Fakultät Maschinenbau
fortschritt studieren

RUHR
UNIVERSITÄT
BOCHUM

RUB

Adjungierte Strömungssimulation und gradientenbasierte Ersatzmodelle in der Turbomaschinenauslegung

Dissertation
zur
Erlangung des Grades
Doktor-Ingenieur

der
Fakultät für Maschinenbau
der Ruhr-Universität Bochum

von

Jan Backhaus

aus Pinneberg

Bochum 2020

Dissertation eingereicht am: 26.05.2020
Tag der mündlichen Prüfung: 25.08.2020
Erstgutachter: Prof. Dr. Reinhard Mönig
Zweitgutachter: Prof. Dr. Hanno Gottschalk

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Die dreidimensionale Strömungssimulation ist aufgrund der steigenden Rechnerleistung zum Standardwerkzeug des industriellen Triebwerksauslegungsprozesses geworden. Durch die stark gesunkenen Laufzeiten können stationäre Simulationen in großen Zahlen durchgeführt werden, um systematische Parameterstudien für Auslegungen durchzuführen. Der hohe technische Reifegrad moderner Triebwerke erfordert jedoch die simultane Abstimmung von immer mehr Parametern, um noch vorhandenes Optimierungspotential ausschöpfen zu können.

Parameterstudien, welche die Interaktion aller Entwurfsvariablen miteinander abbilden können, werden bei den steigenden Parameterzahlen zu aufwändig. Stattdessen gewinnen Optimierungsverfahren an Bedeutung, welche eine gezielte Suche nach einem optimalen Entwurf ermöglichen. Gradientenbasierte Optimierungsverfahren sind effizient für Optimierungsprobleme mit sehr vielen freien Parametern. Sie setzen jedoch voraus, dass Gradienten der Zielfunktionen und Nebenbedingungen bezüglich der Auslegungsparameter effizient berechnet werden können.

Die Idee hinter dieser Effizienzsteigerung kann man auf zwei Weisen motivieren: Zum Einen ist der Gradient die Richtung des steilsten Anstiegs einer Zielfunktion. Zum Anderen beeinhaltet ein Gradient den N-fachen Informationsgehalt eines Funktionswertes, da er aus N partiellen Ableitungen besteht, wobei N die Anzahl Parameter des Optimierungsproblems ist. Mittels finiter Differenzen kann man Gradienten für ein beliebiges Simulationsprogramm berechnen, jedoch sind diese von Genauigkeitsproblemen betroffen und die Berechnungskosten steigen linear mit der Zahl der freien Parameter. Der Vorteil der größeren Informationsmenge wird hierdurch zunichte gemacht.

Es existiert jedoch ein Berechnungsverfahren für Gradienten, das Adjungiertenverfahren, dessen Berechnungskosten unabhängig von der Zahl freier Parameter ist. Eine adjungierte Berechnung liefert den Gradienten für eine Zielfunktion bezüglich beliebig vieler freier Parameter. Die Berechnungskosten hängen bei diesem Verfahren im wesentlichen von der Zahl der gewünschten Zielfunktionen ab. Der Einfluss der Parameterzahl auf den Berechnungsaufwand ist bei dieser Methode vernachlässigbar.

Bei den üblichen Parameterzahlen in der aerodynamischen Detail-Auslegung einer Turbomaschinenstufe im Bereich von 50-200 Parametern und üblicherweise wenigen Zielfunktionen/Nebenbedingungen, lässt der Einsatz adjungierter Methoden erhebliches Beschleunigungspotential erwarten.

Die Forschung der letzten Jahren hat daher zur Entwicklung einiger adjungierter Strömungslöser für aerodynamische Auslegungsoptimierungen, auch im Bereich der Turbomaschinenströmungen, geführt.

Eine noch offene Herausforderung besteht dabei in der Erstellung eines konsistenten adjungierten Lösers zu einem bereits bestehenden Strömungslösers, Testbarkeit dieser Konsistenz, sowie ihr Erhalt unter Fortentwicklungen des Strömungslösers.

Weiterer Forschungsbedarf besteht bezüglich Wahl und Ausgestaltung des Optimierungsverfahrens zur Nutzung adjungiert gewonnener Gradienten.

1.2 Vorgehensweise

Die Arbeit beginnt mit einem Überblick über die für diese Arbeit relevante Forschung. In Kapitel 2 wird das Optimierungsproblem der multidisziplinären Detailauslegung von Turbomaschinenkomponenten formal beschrieben. Daraufhin wird in Kapitel 3 die aerodynamische Zielfunktion in Form der RANS-Gleichungen und ihrer numerischen Lösung dargestellt.

Kapitel 4 beschreibt die Adjungierung iterativer Löser anhand zweier Herleitungsansätze. Daraufhin werden die vier verschiedenen Differentiationstechniken vorgestellt mit denen die Ansätze realisiert werden können.

In Kapitel 5 werden zwei verschiedene Vorgehensmodelle für die Adjungierung eines existierenden Simulationsverfahrens diskutiert, und ihre Anwendung innerhalb des Turbomaschinen-CFD-Verfahrens TRACE dargelegt.

Es folgt der Vergleich zweier Implementierungsweisen in Kapitel 6. Dabei werden zwei prinzipielle Vorgehensweisen unterschieden: Ein Top-Down sowie ein Bottom-Up Ansatz. Die Ansätze werden bezüglich ihrer Auswirkungen auf Entwicklungsaufwand, Wartung und Ressourcenbedarf bewertet.

In Kapitel 7 wird erläutert, wie die Gradienteninformation in die Turbomaschinenoptimierung integriert werden kann. Die hier favorisierte Verwendungsmöglichkeit sind gradientenbasierte Ersatzmodelle. Sie bringen zwar einen deutlichen Mehraufwand bei ihrer Erstellung mit sich, bieten jedoch eine Lösung für einige Probleme der Gradientenabstiegsverfahren. Die hier eingesetzten Kriging Modelle sind in der gradientenfreien Optimierung bereits etabliert. Durch Erweiterung zum gradientenbasierten Kriging kann der Rechenbedarf für die Optimierung gesenkt werden.

Die Einbettung adjungierter Simulationen mit gradientenbasierten Ersatzmodellen in die praktische aerodynamische Turbomaschinenoptimierung stellt jedoch eine Reihe von Herausforderungen an das Adjungiertenverfahren sowie die eingesetzten Optimierungstechniken. Die sich ergebenden Herausforderungen und mögliche Lösungsansätze werden diskutiert.

Den Abschluss der Arbeit bildet Kapitel 8 mit der Anwendung der beschriebenen Techniken auf die Optimierung der gegenläufigen Fan-Stufe CRISPmulti. Hierzu wird eine gradientenfreie Optimierung mit einer gradientenbasierten Optimierung verglichen und die Resultate diskutiert.

1.3 Stand der Forschung

Das Adjungiertenverfahren stammt aus der Theorie der optimalen Steuerung. Diese befasst sich mit der Suche nach einer Funktion als Lösung einer Differentialgleichung, die ein Zielkriterium optimiert.

Lions lieferte die erste Beschreibung wie optimale Steuerungen für Systeme gefunden werden können, deren Zustandsgleichungen partielle Differentialgleichungen sind [1]. Die Übertragung aus dem Gebiet der Regelungstheorie auf die optimale Auslegung aerodynamischer Funktionale wurde zuerst von Pirroneau für die inkompressiblen Euler Gleichungen durchgeführt [2]. Jameson leitete adjungierte Gleichungen für die kompressiblen Euler Gleichungen zur Profiloptimierung her [3] und schlägt die Methode des steilsten Abstiegs als Optimierungsstrategie vor. In weiteren Veröffentlichungen [4, 5] geht er auf die konkrete numerische Implementierung, insbesondere die Diskretisierung, der entstandenen Gleichungen ein und erweitert das Vorgehen auf die reibungsbehafteten Navier-Stokes Gleichungen. Dies stellt Grundlage des Forschungsgebiets adjungierter Strömungslöser dar. Einen guten Überblick des Forschungsstands zu adjungierten CFD Methoden bis 2010 geben ein Buch von Mohammadi und Pironneau [6], sowie ein Überblicksartikel von Peter und Dwight [7].

In der Literatur werden zwei verschiedene Hauptverfahren zur Entwicklung adjungierter CFD Methoden vorgeschlagen: Die Vorgehensweise zunächst die Differentialgleichungen zu adjungieren und das entstandene neue Gleichungssystem zu diskretisieren, wird kontinuierliches Adjungieren genannt. Im Gegensatz dazu steht das diskrete Adjungieren, bei dem eine gegebene Diskretisierung der Differentialgleichungen, in der Regel als implementierter CFD Code, verwendet wird, um den zugehörigen adjungierten Löser zu erzeugen. Der diskrete Ansatz wurde von Giles hinsichtlich Konvergenzeigenschaften untersucht [8]. Gegenüberstellungen der kontinuierlichen und der diskreten Vorgehensweise finden sich in einer Reihe von Veröffentlichungen [9, 10, 11] ohne dass ein Autor die eindeutige Überlegenheit einer Methode feststellt. Diese Arbeit befasst sich ausschließlich mit diskreter Adjungierung.

Kernaufgabe bei der Implementierung eines diskret adjungierten Strömungslösers ist die Generierung adjungierter Routinen für alle berechnungsrelevanten Teile des Ursprungslösers. Zur Differenzierung und Adjungierung von Computerprogrammen existieren vier prinzipielle Verfahren, über die Martins einen systematischen Überblick bietet [12]: Manuelle Differentiation, Bildung von finiten Differenzen, komplexwertige Auswertung zur Bildung finiter Differenzen mit imaginärer Schrittweite sowie algorithmische Differentiation

Müller [13] konstatiert, dass bezüglich der Rechenzeit, die effizientesten adjungierten Routinen durch manuelles Differenzieren erzeugt werden können. Um dem, hiermit verbundenen, hohen Bedarf an Domänenwissen zu begegnen und die Gefahr des Auseinanderdriftens primaler und manuell adjungierter Routinen zu reduzieren, wird die manuelle Differentiation oft mit den anderen Verfahren kombiniert. Hierzu werden entweder finite Differenzen, beispielsweise durch Frey [14], oder die Differentiation mit komplexer Schrittweite [15, 16] verwendet, welche die Schrittweitenwahl vereinfacht, da numerische Auslöschungseffekte reduziert werden. Eine vollständige Eliminierung der Schrittweiten-

problematik kann durch algorithmisches Differenzieren erreicht werden. Umfangreiche Einführungen in diese Thematik findet man bei Griewank und Naumann [17, 18].

Aus praktischen Gründen wird bei der Adjungierung regelmäßig mit Vereinfachungen gearbeitet. Dwight hat eine Reihe verschiedener Vereinfachungen auf den Einfluss der Genauigkeit von Sensitivitäten eines stationären RANS Lösers untersucht und ihre Auswirkungen auf eine konkrete Anwendung dargestellt [19].

Die Anwendung des Rückwärtsmodus algorithmischer Differentiation auf den Quelltext eines Strömungslösers produziert mit geringem manuellen Aufwand einen adjungierten Strömungslöser [20, 21], jedoch bleiben die entstehenden adjungierten Löser auf akademische Anwendungen beschränkt, da Speicher- und Rechenzeitaufwand der entstehenden Löser sehr hoch ausfallen [22]. Kompromisse bieten die Kombination automatisch differenzierter Quelltext-Teile und deren manuelle Differentiation zu einem adjungierten Löser [23, 24, 25]. Dabei kommen sowohl der Vorwärts als auch der Rückwärtsmodus algorithmischer Differentiation zum Einsatz. Während bei diesem Ansatz der Ausgangspunkt das Diskretisierungsverfahren ist, von welchem aus sich hin zur Detailimplementierung vorgearbeitet wird, beginnt eine andere Vorgehensweise mit der vollständigen Differentiation, im Rückwärtsmodus algorithmischer Differentiation um von dort Optimierungen auf höheren Abstraktionsebenen anzuwenden. Dies geschieht in der Regel durch Verbesserung der Algorithmen in AD-Werkzeugen zusammen mit Verbesserungen des primalen Codes [13]. Weitere Löser, die derart differenziert wurden, sind SimpleFoam [21] sowie kürzlich SU2 [26]. Die beiden Ansätze zur Implementation eines adjungierten Lösers wurden im Rahmen dieser Arbeit auch auf das Simulationsverfahren TRACE angewandt [27, 28, 29].

Für die Optimierung der Außenaerodynamik von Flugzeugen beschreiben Leary et al. [30] die verschiedenen Ansätze wie Gradienten eines adjungierten Potentialströmungsverfahrens in eine Optimierung eingebracht werden können: Sie können direkt durch Abstiegsverfahren, in lokalen oder globalen Ersatzmodellen verwendet werden. In der gradientenfreien Optimierung haben sich globale Ersatzmodelle, insbesondere Kriging [31] (z.B. Jones et al. [32], Peter et al. [33], Aulich et al. [34]) als Vorteilhaft erwiesen. Daher liegt es nahe, diese Modelle auch für gradientenbasierte Optimierungen einzusetzen, indem die Gradienten aus dem Adjungiertenverfahren zur effizienteren Informationsgewinnung genutzt werden (Gradient enhanced Kriging, GEK). Chung und Alonso beschreiben dieses Vorgehen für polynomiale Antwortflächen [35] und Kriging [36]. Das in dieser Arbeit verwendete direkte gradientenbasierte Kriging (Direct Gradient Enhanced Kriging, DGEK) wurde zuerst von Morris et al. beschrieben [37]. DGEK wird zunehmend adaptiert (vgl. Beispielimplementierungen zu Forrester et al. [38] und Kriging in Dakota [39]), ist jedoch in den populären Kriging Implementierungen noch nicht enthalten (vgl. Matlab DACE, R DICE Toolbox, Scipy Gaussian Process). Ein wesentlicher Beweggrund für die geringere Verbreitung des DGEK dürfte vor allem darin begründet sein, dass GEK nur in Kombination mit adjungierten Simulationsverfahren einen Vorteil bietet und die numerische Stabilität dieses Verfahrens besonders herausfordernd ist [40]. Die Stabilität des DGEK kann jedoch verbessert werden insbesondere unter Verzicht der strikten Interpolationseigenschaften durch Matrix-Regularisierung (vgl. Abschnitt 3.6 in [38], sowie [40]) bzw. durch Selektion der zum Training des Modells verwendeten Information (vgl. Pi-

voted Cholesky Decomposition in [39]). Ferner ist mit jeder Kriging-Implementierung indirektes GEK durchführbar, welches allerdings numerische Probleme mit sich bringt (vgl. Abschnitt 7.3.3).

Leary et al. [30] sowie Han et. al. [41] demonstrieren die Anwendbarkeit adjungierter CFD Verfahren in Kombination mit gradientenbasiertem Kriging an Optimierungen von 2D Schnitten einer Triebwerksgondel sowie des RAE2822 Gitters.

Für die Optimierung der Beschaufelung einzelner Turbomaschinenstufen sind Campobasso et al. [42], Wu et al. [43], Papadimitrou und Giannakoglou [44] sowie Marta et al. [25] zu nennen. Duta, Shapar und Giles [45] nutzen selektiv AD im Rückwärtsmodus per Source-to-Source Transformation, um einen adjungierten Löser zum CFD Code Hydra zu erzeugen, welchen sie mittels SQP-Verfahren auf den einstufigen transsonischen Testfall NASA Rotor 37 anwenden. Shahpar und Caloni zeigen mit diesem Löser die Optimierung einer Hochdruck-Turbinen Stufe [46] für die Strömungsbedingungen hinter einer Mager-Brennkammer. Hierzu führen sie eine Optimierung ausgehend von der MT1-Turbinenstufe [47] durch, welche sie mit bis zu 273 Parametern in einer Freiform-Deformation mit zusätzlichen Festkörper-Rotationen einzelner Profile variieren. Zielfunktion ist die Erhöhung des Stufenwirkungsgrads, Nebenbedingungen die Einhaltung des Schluckvermögens und des Reaktionsgrads der Turbinenstufe.

Walther und Nadarajah beschreiben einen vollständig manuell differentierten RANS Löser und wenden diesen mittels SQP-Verfahren auf die Optimierung [48] der ersten Stufe des transsonischen Experimentalverdichters THD an, welcher am Verdichterprüfstand der TU-Darmstadt [49] experimentell untersucht wurde [50]. Als Parametrisierung wird eine Hicks-Henne Deformation der gegebenen Oberfläche mit 81 Parametern verwendet. Zielfunktion ist eine Maximierung des isentropen Wirkungsgrads, Nebenbedingungen die Erhaltung des Massenstroms und des Totaldruckverhältnisses.

Wang und He wenden einen kontinuierlich adjungierten Löser und ein nicht näher spezifiziertes Gradientenabstiegsverfahren auf die Optimierung des mehrstufigen „DLR-Verdichters“ an [51, 52].

Die Anwendbarkeit des adjungierten Verfahrens von Frey et al. [14] und die Verifikation der Gradienten und das darauf basierte Training von Kriging Modellen für einen gegenläufigen Fan wurden in [53] demonstriert.

Ein häufiger Ansatz zur Vereinfachung der Adjungierung ist die Annahme, dass die Variablen des Turbulenzmodells konstant bezüglich kleiner Geometrieänderungen ist. Diese Constant-Eddy-Viscosity (CEV) Annahme wurde in einer Reihe von Veröffentlichungen untersucht [54, 19, 55].

Die genannten Publikationen kommen zu dem Ergebnis, dass die Verwendung der CEV Annahme in den betrachteten Optimierungen keinen wesentlichen Einfluss auf das Optimierungsergebnis hatte. Dennoch wird häufig über die Implementierung adjungierter Turbulenzmodelle publiziert: Das Baldwin-Lomax-Modell durch Kim und Kim [56], das Spalart-Allmaras Modell durch Nielsen und Anderson [57] das Wilcox k-ω Modell durch Giles, Duta und Müller [58] sowie im Bereich kontinuierlich adjungierter Löser das k-ω-SST Modell durch Kavvadias et al. [59]. Die Verwendung von Transitionsmodellierung in adjungierten Simulationen beschränkt sich im wesentlichen auf die Optimierung von Laminarprofilen in der Außenaerodynamik. Beispiele sind die Adjungierung des e^N

Ansatzes durch Lee und Jameson [60] und des γ-Re_θ-Modells durch Khayatzadeh und Nadarajah [61].

Neben der Anwendung zur Gradientenberechnung in der Optimierung haben adjungierte Löser auch Anwendung in der Quantifizierung von Unsicherheiten (Uncertainty Quantification) für Simulationsergebnisse gefunden. Es sind in letzter Zeit insbesondere die Auswirkung von Abweichungen in der Beschaufelungsform von der angenommenen Form betrachtet worden. Beispielsweise betrachten Giebmanns et al. [62, 63, 64] die Auswirkung von betriebsbedingten Abnutzungserscheinungen von Triebwerksbeschaufelung auf die Leistungsfähigkeit der Komponente. Fertigungsbedingte Formabweichungen von Turbinenschaufeln wurden mit adjungierten Methoden aerodynamisch bewertet durch Zamboni [65] und Liefke [66, 67]. Überblicksarbeiten zum Stand der Bewertung von Fertigungs- und Abnutzungsbedingter Formabweichung von Gasturbinen und Flugtriebwerken erstellten Montomoli et al. [68] und Massini [69].

Für solche Betrachtungen muss jedoch immer im Einzelfall gerechtfertigt werden, dass die betrachteten Abweichungen sich noch im Bereich linearer Abhängigkeit der unsicheren Größen befinden. Giebmanns et al. stellen dies durch Schrittweitenstudien mit nichtlinearen Rechnungen parameterweise sicher. Zamboni et al. vergleichen die erhaltenen Ergebnisse gegen die Resultate aus Monte-Carlo Experimenten. Dwight und Han demonstrieren die Kombination von lokaler Sensitivitätsinformation mit globalen Ersatzmodellen zur Unsicherheitsschätzung [70].

Es existieren also bereits industriell einsetzbare adjungierte Löser sowie Anwendungen auf relevante Entwurfsprobleme. Wenig behandelt wurden bisher die Themenfelder Modellumfang und Konsistenzerhaltung des adjungierten Lösers, Robustheit des Optimierungsprozesses gegen Fehlschlag des Bewertungsprozesses, sowie Kopplung differenzierter und nicht-differenzierter Zielfunktionale und Nebenbedingungen. Sie sollen daher in der vorliegenden Arbeit thematisiert werden.

1.4 Beitrag der Arbeit zum Stand der Forschung

Im ersten Teil dieser Arbeit wird die Vorgehensweise zur Implementierung eines diskret adjungierten Strömungslösers genauer betrachtet. In der existierenden Literatur wird bezüglich Implementierungstechnik hauptsächlich der Unterschied zwischen kontinuierlicher und diskreter Adjungierung diskutiert. Für die diskrete Adjungierung können jedoch auch unterschiedlich vorgegangen werden und dies hat dann Auswirkungen auf Verifizierbarkeit, Wartbarkeit, Erweiterungsaufwand und Ressourcenbedarf des diskret adjungierten Lösers. Um verschiedene Ansätze vergleichen zu können werden diese zunächst systematisiert: Zum Einen können vier verschiedene Differentiationstechniken verwendet werden, und zum Anderen können diese auf verschiedene Abstraktionsebenen des existierenden Simulationsverfahrens angewandt werden. In der vorliegenden Arbeit wird eine mögliche Einteilung der Abstraktionsebenen vorgeschlagen: Die Erhaltungsgleichungen, ihre Diskretisierung, das numerische Lösungsverfahren sowie dessen Implementierung als Computerprogramm. Es werden zwei prinzipielle Vorgehensweisen unterschieden: Ein Top-Down Ansatz, welcher mit manueller Adjungierung auf Ebene der diskretisierten Erhaltungs-

gleichungen beginnt, um dann manuell, unter Zuhilfename von Finiten-Differenzen oder algorithmischer Differentiation die benötigten Teile des Quelltexts zu differenzieren. Diesem wird ein Bottom-Up Ansatz gegenübergestellt, welcher zunächst den vollständigen Quelltext algorithmisch differenziert und daraufhin Kenntnisse über Lösungsverfahren und zugrundeliegende Gleichungen zur Beschleunigung des entstandenen adjungierten Lösers verwendet.

Für beide Ansätze finden sich bereits Umsetzungsbeispiele in der Literatur. Die Umsetzungen wurden jedoch noch nicht vergleichend untersucht und bisher war dies auch nicht gut möglich, da verschiedene Strömungslöser mit den verschiedenen Ansätzen behandelt wurden. In der vorliegenden Arbeit werden Top-Down und Bottom-Up Ansatz auf den gleichen Strömungslöser angewandt und bezüglich praxisrelevanter Kriterien verglichen. Es wird gezeigt, dass eine Bottom-Up Vorgehensweise für laufend weiterentwickelte Simulationsprogramme bezüglich Funktionsumfang, Verifizierbarkeit und Wartbarkeit vorteilhaft ist, und sich ein Nachteil im Bedarf an Rechenressourcen durch manuelle Optimierung des Lösers weitgehend ausgleichen lässt.

Ein weiterer Beitrag der Arbeit betrifft die Berechnung von Gradienten in industrierelevanten Auslegungsprozessen. Die Arbeit legt dar, welche Konsequenzen dies für den adjungierten Löser hat, welche Teile des Bewertungsprozesses auf welche Art differenziert werden können und wie ein adäquater adjungierte Bewertungsprozess konstruiert werden kann.

Der dritte Beitrag ist die Diskussion auf welche Art die adjungiert erhaltenen Gradienten in der Optimierungsiteration verwendet werden können. Es wird argumentiert, dass gradientenerweiterte Ersatzmodelle für die vorgestellte Problemklasse vorteilhaft gegenüber der direkten Verwendung in Abstiegsverfahren sind. Gradientenerweitertes Kriging wurde bereits früher im Kontext adjungierter Strömungssimulation beschrieben, hier wird darüber hinaus gezeigt wie in Mehrzieloptimierungen in Zusammenwirkung mit nichtdifferenzierten Zielfunktionalen und bei Fehlschlag einzelner Simulationsteile ein fehlertoleranter Optimierungsprozess gestaltet werden kann. Darüberhinaus wird ein Beitrag zur Stabilisierung gradientenerweitertem Krigings durch eine neue Regularisierungsmethode vorgestellt.

Durch die Anwendung auf eine gegenläufige Fan-Stufe wird die Funktionsfähigkeit der beschriebenen Techniken in einer technisch relevanten Optimierung gezeigt.

1.5 Implementierungen und Kooperationen im Rahmen der Arbeit

Die Implementierung moderner Strömungssimulationsverfahren und die darauf aufbauenden Auslegungsmethoden sind ohne die Kooperation vieler Personen und Institutionen nicht mehr denkbar. Diese Arbeit baut auf Algorithmen und Software auf, die teilweise von anderen übernommen und fortentwickelt oder gemeinschaftlich entwickelt wurde. Im Folgenden sollen die Beiträge detaillierter erläutert werden um die Arbeiten des Autors von denen anderer Personen trennen zu können.

Der nichtlineare stationäre Strömungslöser aus dem Simulationspaket TRACE ist Er-

gebnis vieler Forschungsarbeiten und stand zur Beginn der Arbeit zur Verfügung. Für Literaturangaben siehe Kapitel 3. Im Rahmen der Arbeit wurden Verbesserungen zum nichtlinearen Löser beigesteuert.

Die Top-Down Adjungierung dieses Lösers ist die Leistung von Frey et al. [14]. Im Rahmen dieser Arbeit wurde die Anwendbarkeit des Lösers in der Turbomaschinenoptimierung mit ersatzmodellbasierter Optimierung demonstriert [53]. Es wurde die Adjungierung der nicht 1-zu-1 Schnittstellen beigesteuert. Weiterhin fand eine Untersuchung des Einflusses der zuerst verwendeten finiten Differenzen auf die Ergebnisgenauigkeit statt [71]. Daraufhin wurde einen Ansatz konzipiert und umgesetzt mittels dessen algorithmisches Differenzieren im Vorwärtsmodus ohne Leistungseinbuße und große Umstrukturierungen umgesetzt werden kann [28].

Die Implementierung des Bottom-Up Ansatzes entstand im Projekt R&E-TURB innerhalb des BMWi Luftfahrtforschungsprogramms IV sowie im Projekt 2-Shaft-Compressor innerhalb des EU Rahmenprogramms CleanSky 2. Sie ist das Ergebnis der Kooperation des Lehrstuhl SciComp der TU-Kaiserslautern sowie der MTU-Aero-Engines mit dem DLR Institut für Antriebstechnik. Die Werkzeuge zur algorithmischen Differenzierung und adjungierter Kommunikation wurden vom Lehrstuhl SciComp der TU-Kaiserslautern bereitgestellt und entwickelt. Der Autor dieser Arbeit hat zusammen mit den Kooperationspartnern an Konzeption, Vorarbeiten am primalen Löser, Implementierung des adjungierten Lösers, Verifizierung, Fehlersuche und Test sowie der Implementierung von Effizienzsteigerungsmaßnahmen durchgeführt.

Das Optimierungswerkzeug AutoOpti wurde von Voß et al. [72] entwickelt und hier unverändert übernommen.

Die Implementierung des gradientenerweiterten Krigings wurde von Schmitz [73] übernommen. Es wurde im Rahmen der Arbeit eine neue Regularisierungstechnik für die gradientenerweiterte Korrelationsmatrix beigetragen.

Der gradientenfreie Bewertungsprozess wurde von Lengyel et al. [74] bereitgestellt. Im Rahmen dieser Arbeit wurde hierauf basierend eine Prozesskette entwickelt, welche wahlweise Gradienten zu den berechneten Funktionswerten bereitstellt. Hierzu wurde ein differenzierter Prä-Prozesses erstellt und mit dem ebenfalls neuentwickelten adjungierten RANS-Löser sowie adjungierten Post-Processings zu einer Gesamtprozesskette integriert, getestet und an das Optimierungsverfahren AutoOpti angebunden.

2 Turbomaschinenauslegung durch Optimierung

2.1 Überblick

Die simulationsbasierte Optimierung einer Turbomaschine ist im mathematischen Sinn die Minimierung mehrerer Funktionen unter Nebenbedingungen. Allerdings sind Zielfunktionen und Nebenbedingungen nicht analytisch gegeben, sondern durch eine Kette von Computerprogrammen.

Ein Teil dieser Arbeit beschreibt Verfahren um von aerodynamischen Zielfunktionen und Nebenbedingungen auch partielle erste Ableitungen dieser Bewertungsketten berechnen zu können. Um mittels Optimierungsverfahren neue Entwürfe zu generieren, die wiederum bewertet werden. Diese Struktur ist in Abbildung 2.1 dargestellt.

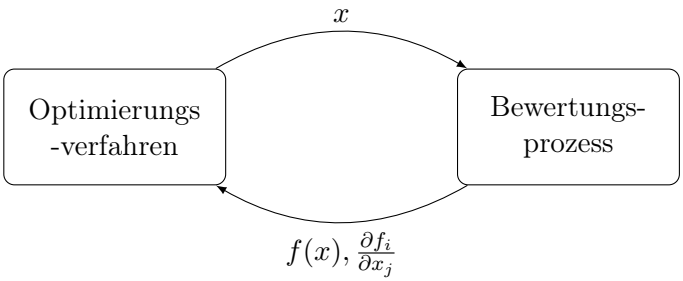


Abbildung 2.1: Grundschema der CFD basierten Optimierung

Ein generischer Optimierungsalgorithmus erzeugt dabei einen Vektor von Gleitkommazahlen aus dem Entwurfsraum $x \in D \subset \mathbb{R}^N$. Dieser wird vom Bewertungsprozess als Beschreibung einer Turbomaschinenkomponente interpretiert, woraufhin diese einen Vektor von Bewertungsgrößen, also Zielfunktions- und Nebenbedingungswerte, $f(x)$ als Ergebnis einer numerischen Simulation generiert. In dieser Arbeit ist der Kern des Bewertungsprozesses eine dreidimensionale aerodynamische Strömungssimulation (Computational Fluid Dynamics, CFD).

Diese Ergebnisse werden an den Optimierungsprozess zurückgemeldet, woraufhin dieser versucht, einen besseren Vektor von Entwurfsvariablen zu finden.

Ziel dieser Arbeit ist die Berechnung von Ableitungen $\partial f_i / \partial x_j$ der aerodynamischen Bewertungsfunktion für die Turbomaschinenauslegung und deren Verwendung im Optimierungsverfahren.

2.2 Charakterisierung des Optimierungsproblems

Die Optimierungsprobleme welche in der vorliegenden Arbeit behandelt werden, sind durch folgende Eigenschaften gekennzeichnet: Es wird mindestens eine Stufe einer Turbomaschine dreidimensional simuliert und ausgewählte Oberflächen mittels Entwurfsparametern veränderbar gehalten. Zur Zeit kommen hierfür bis 200 kontinuierliche Parameter zum Einsatz. Diskrete Parameter, wie beispielsweise die Schaufelzahl kommen in diesem Stadium des Entwurfs normalerweise nicht mehr vor, da sie typischerweise im Vorentwurf festgelegt werden.

Die Zielfunktionen werden als kontinuierlich, nichtlinear und mindestens einmal stetig differenzierbar angenommen. Die zu optimierenden Zielfunktionen sind Ergebnisse eines Simulationsprozesses, weshalb jede Funktionsauswertung mit hohem Verbrauch an Rechenressourcen (Speicher, CPU) verbunden ist und auch auf Höchstleistungsrechnern Minuten bis Stunden benötigt. Wir sagen daher, dass die Zielfunktion teuer ist. Ferner ist es möglich, dass die Zielfunktion nicht an jeder Stelle des Parameter-Raums ausgewertet werden kann, da Teile des Simulationsprozesses fehlschlagen können. Wir gehen davon aus, dass Ableitungen der aerodynamischen Zielfunktionen zu vergleichbaren Kosten wie die Zielfunktion selbst ausgewertet werden können. Es müssen Nebenbedingungen eingehalten werden, welche geometrischer, festigkeitsmechanischer oder aerodynamischer Art sind und teilweise ebenfalls durch Simulationen gegeben sind. Es liegen nicht immer für alle Nebenbedingungen adjungierte Verfahren vor.

Es ist davon auszugehen, dass insbesondere die aerodynamischen Zielfunktionen multimodal sind, also mehrere Optima besitzen. Sicher ist, dass aufgrund numerischer Effekte kleinere lokale Minima existieren (vgl. hierzu im Kontext der Außenaerodynamik Bons et al. [75]).

2.3 Mehrzieloptimierung

Häufig gibt es zwei oder mehr konkurrierende Ziele für die ein Kompromiss aus einer Menge Pareto–optimaler Lösungen gefunden werden soll. Ein Beispiel aus der Turbomaschinenauslegung sind die Ziele Steigerung des Wirkungsgrads und Erweiterung des Betriebsbereichs. Es ist bekannt, dass nicht beide Ziele mit einem Entwurf gleichzeitig optimal zu erfüllen sind. Optimierungen, in denen mehr als eine Zielfunktion gleichzeitig optimiert werden sollen, heißen Mehrzieloptimierung.

Die Zielfunktion kann dann als Vektor–Abbildung geschrieben werden: $\mathbf{f} : \mathbb{R}^N \mapsto \mathbb{R}^M$. Mit N wird die Dimension des Designraums und mit M die Dimension des Zielfunktionsraumes bezeichnet.

Bei eindimensionalen Zielfunktionen kann ein lokales Optimum besser, gleichwertig oder schlechter als ein anderes lokales Optimum sein. Es gibt nur einen global optimalen Funktionswert. Optimiert man multikriteriell, also mehrere Zielfunktionen gleichzeitig, wird der Zielfunktionsraum mehrdimensional und man verliert diese strenge Ordnungsrelation: Verschiedene Lösungen können trotz unterschiedlicher Zielfunktionswerte global

optimal sein. Anders gesagt kann eine Lösung auch dann ein globales Optimum sein, wenn sie in einem, oder mehreren Zielfunktionswerten schlechter ist als eine andere Lösung.

Als Ordnungsrelation ist in multikriteriellen Optimierungen das Pareto–Kriterium gebräuchlich:

Eine Lösung $y = (y^{(1)}, ..., y^{(M)})$ dominiert eine andere Lösung $z = (z^{(1)}, ..., z^{(M)})$ wenn

$$\forall i = 1, 2, \ldots, M \quad y^{(i)} \geq z^{(i)} \tag{2.1}$$

und

$$\exists j = 1, 2, \ldots, M \quad y^{(j)} > z^{(j)}. \tag{2.2}$$

Eine Lösung heißt Pareto-optimal, wenn keine Lösung existiert, die sie dominiert.

Bezeichnet man mit D die Anzahl Lösungen, die eine Lösung y dominieren, dann ist der Pareto–Rang dieser Lösung $1 + D$. Die Menge aller Pareto-optimaler Punkte eines Optimierungsproblems wird auch Pareto-Front genannt. Eine detaillierte Diskussion der Begriffe findet sich bei Nocedal et al. [76].

Zur Suche solcher Pareto-Optimaler Punkte wird gelegentlich das mehrdimensionale Problem auf ein eindimensionales reduziert. Dies geschieht Beispielsweise durch Skalarisierung $f \cdot w$ mit im Vorhinein festzulegenden Gewichten $w \in \mathbb{R}^M$. Für eine Wahl der Gewichte wird jedoch höchstens eine Pareto-optimale Lösung gefunden und selbst bei Variation der Gewichte können im Allgemeinen nicht alle Pareto-optimalen Punkte gefunden werden.

Ziel der multikriteriellen Optimierung ist das Auffinden möglichst vieler Pareto-optimaler Lösungen die den gesamten Wertebereich der Pareto-Front oder gewünschte Teilbereiche möglichst gleichmäßig abdecken.

2.4 Wahl der Optimierungsmethode

Ein bekanntes Theorem der Optimierungs-Literatur ist das „No-Free-Lunch-Theorem“ [77]. Es besagt, dass kein Optimierungsverfahren in jeder Klasse von Optimierungsproblemen allen anderen Verfahren überlegen sein kann. Es ist daher notwendig, das Optimierungsverfahren der Problemklasse entsprechend auszuwählen. Da die Zielfunktionsauswertungen teuer sind ist ein wesentliches Auswahlkriterium die Anzahl nötiger Bewertungen.

Diese Arbeit befasst sich mit Optimierungsverfahren, die auf kontinuierlichen Variablen definiert sind. Ein wesentliches Unterscheidungsmerkmal für kontinuierliche Optimierungsverfahren ist, ob ein Verfahren Gradienten der Zielfunktion verwenden kann. In der Literatur wird eine Vielzahl von gradientenfreien [78], sowie diverse gradientenbasierte [76] Optimierungsverfahren beschrieben.

Ein sehr häufig anzutreffendes gradientenfreies Optimierungsverfahren sind evolutionäre Algorithmen [79]. Die wichtigsten Vertreter der gradientenbasierten Optimierungsmethoden sind Gradienten-Abstiegsverfahren und Trust-Region Methoden (vgl. Kapitel 3 und 4 in [76]). Zingg [80] demonstriert im Vergleich zwischen Gradienten-Abstiegsverfahren

und evolutionären Algorithmen, dass für die aerodynamische Optimierung gradientenbasierte Verfahren erheblich weniger Zielfunktionsauswertungen zur Konvergenz der Optimierung benötigen.

In der simulationsbasierten aerodynamischen Optimierung haben sich dennoch gradientenfreie Verfahren etabliert, da Gradienten des Optimierungsproblems für lange Zeit nicht effizient berechnet werden konnten. Zur Reduktion der Anzahl von Zielfunktionsauswertungen setzt man hier auf Ersatzmodellierung [38]. Als einer der populärsten Ansätze in der simulationsbasierten Optimierung hat sich Kriging [31] etabliert. Für eine Erläuterung von Ersatzmodellierung und Kriging sei auf Abschnitt 7.1 verwiesen.

Ziel dieser Arbeit ist die Beschleunigung von Optimierungen durch die Bereitstellung von Gradienteninformation. Der naheliegende Ansatz wäre die Verwendung von Gradienten-Abstiegsverfahren. Folgende, im vorigen Abschnitt dargestellte, Eigenschaften des Optimierungsproblems erschweren dies jedoch:

1. Fehlen effizienter Gradientenberechnungsverfahren für einen Teil der Nebenbedingungen
2. Nicht-Differenzierbarkeit einzelner Nebenbedingungen
3. Mögliches Fehlschlagen der simulationsbasierten Prozesse
4. Lokale Minima der Zielfunktion
5. Die Erzeugung verschiedener Pareto-Optimaler Lösungen in Mehrzieloptimierungen

Lösungen für diese Herausforderungen sind zum Teil Gegenstand aktueller Forschung. Um dennoch die Gradienteninformation nutzen zu können, wird in dieser Arbeit der Ansatz verfolgt die bereits etablierte ersatzmodellbasierte Optimierung durch Gradienten des Designproblems zu beschleunigen. Dies kann durch gradientenbasierte Ersatzmodelle erreicht werden. In diesen Modellen werden nicht nur die Werte der Zielfunktion einbezogen, sondern zusätzlich auch der Gradient der Zielfunktion. Da ein Gradient einer Funktion von N unabhängigen Größen, N partielle Ableitungen enthält, bekommt man durch das Adjungiertenverfahren die N-fache Informationsmenge einer Zielfunktionsauswertung bei vergleichbaren Rechenkosten. Die Gradienten dienen bei diesem Ansatz also der Einsparung von Zielfunktionsberechnungen für die Ersatzmodellierung. Er ermöglicht es die etablierten Ersatzmodelle beizubehalten und den Aufwand für ihre Erstellung zu reduzieren. Eine Abschätzung über die zu erwartende Beschleunigung erhalten wir unter der Annahme, dass für alle M Zielfunktionen und Nebenbedingungen Gradienten adjungiert berechnet werden, diese Berechnungen alle gleich aufwändig sind und der Aufwand für die Erstellung und Optimierung des Kriging-Modells vernachlässigt werden kann.

Die maximal mögliche Beschleunigung durch das Adjungiertenverfahren ist $\frac{N}{M}\frac{T_{\text{primal}}}{T_{\text{adjoint}}}$.

Darin sind T_{primal} und T_{adjoint} die Laufzeit des primalen respektive adjungierten Lösers.

In der Realität wird dieses Maximum an Beschleunigung nicht erreicht. Einflüsse auf die Beschleunigung durch das Adjungiertenverfahren haben die Glattheit der Zielfunktionen

und Nebenbedingunen, ob es mehrere lokale Minima gibt und der Start der Optimierung sich im Einzugsbereich dieses befindet und ob ein globaler Trend der Zielfunktion vorhanden ist. Außerdem wird die Beschleunigung durch die Genauigkeit der iterativ berechneten Gradienten und den Umgang des Optimierungsverfahrens mit den Gradienten beeinflusst.

Dennoch sieht man, dass die Einführung von Gradienten in das Optimierungsverfahren dann vorteilhaft ist, wenn das Optimierungsproblem deutlich mehr Parameter als Zielfunktionen und Nebenbedingungen aufweist $N \gg M$ und T_{adjoint} möglichst klein ist. Mit der detaillierten Darstellung des Kriging-Verfahrens wird klar, dass die Zeit für die Erstellung eines Kriging Modells kubisch mit der Zahl der Trainingspunkte, bei gradientenerweitertem Kriging auch mit der Entwurfsraumdimension ansteigt und somit die Vernachlässigung des Ersatzmodellierungsaufwands ab einer gewissen Zahl von Optimierungsparametern nicht mehr gilt. Für Optimierungsprobleme mit einem Entwurfsraum welcher um Größenordnungen mehr Dimensionen besitzt, müsste die Wahl des Optimierungsverfahrens erneut betrachtet werden.

3 Strömungssimulation

Im Folgenden werden die zeitgemittelten Navier-Stokes Gleichungen beschrieben, da sie in dieser Arbeit zur aerodynamischen Bewertung verwendet werden. Ein Teil des Turbomaschinen-CFD-Verfahrens TRACE dient der diskreten Lösung dieser Gleichungen. Auf diesem bauen die Verfahren dieser Arbeit auf, seine eigentliche Implementierung wird als gegeben betrachtet und nur überblicksartig skizziert. Genauere Beschreibungen der numerischen Methoden finden sich bei Eulitz [81] Nürnberger [82] und Kügeler [83].

3.1 Wahl der Modellgleichungen

Das aerodynamische Verhalten einer Turbomaschinekomponente wird durch die Navier-Stokes Gleichungen beschrieben. Da man im allgemeinen keine analytische Lösung dieser Gleichungen angeben kann, werden diese Gleichungen diskretisiert und ihre Lösung numerisch approximiert. Die Strömung in Turbomaschinen ist aufgrund inhärenter instationärer Effekte sowie der Durchströmung rotierender und stehender Schaufelreihen grundsätzlich instationär und kann exakt nur durch instationäre Simulation wiedergegeben werden. Trifft man keine vereinfachenden Annahmen bezüglich turbulenter Vorgänge führt dies zur Direct Numerical Simulation (DNS). Die benötigte Netzmaschengröße und Zeitschrittweite zur Auflösung aller turbulenter Strukturen beschränkt diese Art der Simulation bisher auf Forschungsprojekte auf kleinen Kontrollvolumina bei moderaten Reynoldszahlen (bspw. bei Wheeler et al. [84]). Für die Reynoldszahlen technisch relevanter Turbomaschinen, der verfügbaren Rechenkapazität sowie der Anzahl zu bewertender Designs ist man für Auslegungen auf absehbare Zeit auf vereinfachende Annahmen angewiesen [85]. Zu einer erheblichen Reduktion des Rechenaufwands führen dabei gröbere Rechennetze, welche nicht mehr alle relevanten Turbulenzstrukturen auflösen sondern die Effekte vernachlässigter Wirbelgrößen durch ein numerisches Modell approximieren. Zunehmend relevant werden hierbei Large-Eddy Simulationen (LES) [86], deren Durchlaufzeit für relevante Anwendungen jedoch noch im Bereich von Wochen liegt. Durch Reynolds- oder Favre-Mittelung der Navier-Stokes Gleichungen erhält man die Reynolds-Averaged-Navier-Stokes (RANS) für die eine Vielzahl von Turbulenzmodellen entwickelt wurde. Dazwischen existiert eine Vielzahl hybrider LES-RANS-Verfahren [86].

Eine erhebliche Verkürzung der Rechenzeit und des Speicherbedarfs wird durch zeitliche Mittelung der Gleichungen und Beschränkung auf stationäre Strömungseffekte erreicht. Hierbei wird angenommen, dass eine vollständige Ausmischung der Strömung in Umfangsrichtung zwischen rotierenden und stehenden Reihen erfolgt. Diese Annahme erlaubt es mittels einer Mischungsebene (Mixing-Plane, s. [87]) einen in Umfangsrichtung konstanten Strömungszustand zwischen Schaufelreihen zu erzwingen. Daher kann man die Umfangssymmetrie der verschiedenen Reihen ausnutzen und stationäre Simulationen

mit je einer Schaufelpassage pro Reihe durchführen. Die Leistungsfähigkeit heutiger Rechnersysteme hat die Berechnungszeit für stationäre Simulationen derart verkürzt, dass sie sich als Bewertungswerkzeug in der Detailoptimierung von Turbomaschinenkomponenten eignen.

Da die Vernachlässigung sämtlicher instationärer Effekte gelegentlich eine zu starke Vereinfachung darstellt, kann eine Transformation der instationären Gleichungen in den Frequenzbereich [88] erfolgen, was als Harmonic-Balance-Verfahren (HB) [89, 90] im vorliegenden Löser umgesetzt wurde [91, 92]. Dies erlaubt die Beschränkung auf ausgewählte periodische Phänomene, was erhebliche Einsparungen der Rechenzeit gegenüber vollständig instationärer Simulationen ermöglicht. Die in dieser Arbeit beschriebenen Adjungierungsverfahren wurden durch Engels-Putzka aufgegriffen und auf das Harmonic-Balance Verfahren erweitert [93, 94, 95].

3.2 Grundgleichungen

Die kompressiblen Navier-Stokes Gleichungen im rotierenden Referenzsystem haben die Form

$$\frac{\partial \mathbf{q}}{\partial t} + \operatorname{div}\left(\mathbf{F}_c(\mathbf{q}) - \mathbf{F}_v(\mathbf{q})\right) + \mathbf{S}(\mathbf{q}) = 0. \tag{3.1}$$

Wobei $\mathbf{q}$ der Vektor konservativer Variablen ist und $\mathbf{F}^c$ sowie $\mathbf{F}^v$ die Vektoren der konvektiven und viskosen Flüsse bezeichnen:

$$\mathbf{q} = \begin{bmatrix} \rho \\ \rho U_x \\ \rho U_y \\ \rho U_z \\ \rho E \end{bmatrix}, \mathbf{F}^c = \begin{bmatrix} \rho\mathbf{U} \\ \rho U_x \mathbf{U} + p\mathbf{e}_x \\ \rho U_y \mathbf{U} + p\mathbf{e}_y \\ \rho U_z \mathbf{U} + p\mathbf{e}_z \\ \rho \mathbf{U} H_\mathrm{t} \end{bmatrix}, \mathbf{F}^v = \begin{bmatrix} 0 \\ \boldsymbol{\tau}_x \\ \boldsymbol{\tau}_y \\ \boldsymbol{\tau}_z \\ \boldsymbol{\tau}\mathbf{U} + \mathbf{Q} \end{bmatrix}. \tag{3.2}$$

Darin ist ρ die Dichte und $\mathbf{U} = (U_x, U_y, U_z)$ der Geschwindigkeitsvektor. Die Rothalpie ist $H_\mathrm{t} = e + \frac{1}{2}\left(\|\mathbf{U}\|^2 - (|\omega| r)^2\right) + \frac{p}{\rho}$ mit dem Radius r und der Winkelgeschwindigkeit ω. Für die innere Energie des Gases wird ein ideales Gas angenommen:

$$e = \frac{1}{\gamma - 1} R_s T, \quad p = \rho R_s T. \tag{3.3}$$

Dabei bezeichnet R_s die spezifische Gaskonstante, T die Temperatur und γ den Isentropenexponenten. Der viskose Spannungstensor τ lautet mit der Stokes'scher Hypothese für ein Newton'sches Fluid

$$\tau_{ij} = 2\mu \left(s_{ij} - \frac{1}{3} \operatorname{Tr}(\mathbf{s}) \delta_{ij} \right) \tag{3.4}$$

und

$$s_{ij} = \frac{1}{2} \left(\frac{\partial \mathbf{U}_j}{\partial x_i} + \frac{\partial \mathbf{U}_i}{\partial x_j} \right). \tag{3.5}$$

Darin bezeichnet μ die dynamische Viskosität. Die Wärmeflüsse werden nach dem Fourier'schen Gesetz berechnet

$$\mathbf{Q} = -\kappa \nabla T. \tag{3.6}$$

Der Vektor der Quellterme beinhaltet den Beitrag der Coriolis– und Zentrifugalkraft auf die Impulserhaltung:

$$\mathbf{S} = \begin{bmatrix} 0 \\ 0 \\ -2\rho\omega U_z - \rho|\omega|^2 r\mathbf{e}_r \\ 2\rho\omega U_y - \rho|\omega|^2 r\mathbf{e}_r \\ 0 \end{bmatrix} \tag{3.7}$$

Der Einfluss der Temperatur auf die dynamische Viskosität wird durch das Sutherland–Gesetz approximiert. Da sich technisch relevante Strömungen unter der verfügbaren Rechenleistung nicht fein genug diskretisieren lassen, um sie mit den oben stehenden Gleichungen zu lösen, wendet man Favre- und Reynoldsmittelung auf die oben stehenden Gleichungen an. Unabhängig von der Art der Mittelung nennt man diese Gleichungen Reynolds-Averaged-Navier-Stokes (RANS). Dabei werden alle zeitlich variablen Größen in einen stationären und fluktuierenden Teil zerlegt

$$\phi(q_i, t) = \bar{\phi}(q_i) + \phi'(q_i, t) \tag{3.8}$$

wobei nur der stationäre Teil, welcher den gemittelten Größen entspricht, simuliert wird. Da nichtlineare Terme fluktuierender Größen durch die Mittelung nicht wegfallen, müssen diese modelliert werden. Insbesondere ist die Mittelung des Reynolds–Spannungs–Tensor

$$\boldsymbol{\tau}^F = -\bar{\rho}\overline{\mathbf{U}\mathbf{U}} \tag{3.9}$$

zu modellieren. Hierzu wurde eine große Zahl von Turbulenzmodellen entwickelt und beschrieben. Diese reichen von algebraischen Modellen über Zwei–Gleichungs–Wirbelviskositätsmodellen bis zur explizit algebraischen oder differentiellen Modellierung des gesamten Reynolds–Spannungs–Tensors. Für eine eingehende Darstellung hierzu siehe [96]. In dieser Arbeit wird hauptsächlich das Wilcox k–ω Modell [97] mit Modifikationen eingesetzt. Die Modifikationen korrigieren die Staupunkanomalie [98] und modellieren Rotations–Effekte [99].

Für die Modellierung des laminar–turbulenten Umschlags in Strömungen stehen verschiedene Transitionsmodelle zur Verfügung, wobei im Rahmen dieser Arbeit das Multimode–Modell [100] sowie das γ–Re_θ–Modell [101] getestet wurden.

3.3 Diskretisierung und Lösungsverfahren

Für die approximative Lösung der Gleichungen werden diese über ein Kontrollvolumen integriert, welches im rotierenden Relativsystem feststeht, und der Gauß'sche Integralsatz angewendet. Diese Integralform wird mittels des Finite–Volumen Verfahrens diskretisiert (für Details der Umsetzung vgl. [81]). Die verwendeten Netze können aus strukturierten

und unstrukturierten Blöcken bestehen [102]. Die so diskretisierte Form der Navier-Stokes Gleichungen hat die Form

$$R_i = V_i^{-1} \sum_j (F_j^c - F_j^v) - S_i. \tag{3.10}$$

Wobei R den Vektor der Residuen, also den zu reduzierenden Lösungsfehler, in der i–ten Zelle bezeichnet, V_i deren Volumen, F_j den Flussvektor durch die j–te Zellfläche und S_i die Quellterme bezeichnen. Die konvektiven Flüsse werden durch Upwind–Diskretisierung mittels Roe's Riemann Löser [103] und van Leers MUSCL-Rekonstruktion [104] gelöst. Zur Vermeidung unphysikalischer Oszillationen hinter Stößen werden Limiter Funktionen eingesetzt. Im Hinblick auf linearisierte und adjungierte Simulation hat sich dabei ein van–Albada–artiger Limiter als vorteilhaft herausgestellt [71]. Durch Hartens Entropie–Korrektur [105] wird die Entstehung unphysikalischer Diskontinuitäten aufgrund der Verletzung der Entropiebedingung verhindert. Die viskosen Flüsse werden mittels eines zentralen Schemas zweiter Ordnung diskretisiert.

Die Gleichungen werden mittels eines impliziten Pseudo–Zeitschrittverfahrens mit lokaler Einstellung der Zeitschrittweite iterativ approximiert:

$$\Delta \mathbf{q}^{(n)} = \mathbf{q}^{(n+1)} - \mathbf{q}^{(n)} \tag{3.11}$$

Zur Bestimmung von $\Delta \mathbf{q}^{(n)}$ nutzt man das lineare Gleichungssystem

$$\left(\Delta \boldsymbol{\tau}^{-1} + \left. \frac{\partial \widetilde{\mathbf{R}}}{\partial \mathbf{q}} \right|_{\mathbf{q}^{(n)}} \right) \Delta \mathbf{q}^{(n)} = \mathbf{R}^{(n)}. \tag{3.12}$$

Wobei $\Delta \boldsymbol{\tau}$ die durch das Courant-Friedrichs-Levy (CFL) Kriterium bestimmten lokalen Zeitschrittweiten sind. $\frac{\partial \widetilde{\mathbf{R}}}{\partial \mathbf{q}}$ ist die Jacobi–Matrix von $\widetilde{\mathbf{R}}$, einer Approximation erster Ordnung des Residuums.

Die Lösung wird dabei iterativ angenähert; Entweder mittels symmetrischen Gauß–Seidel Algorithmus (SGS), oder der symmetrischen sukzessiven Relaxationsverfahrens (SSOR). Eine detaillierte Darstellung des verwendeten impliziten Lösungsverfahrens findet sich bei Nürnberger [82].

3.4 Randbedingungen

Die Randbedingungen sowie die Kopplung von Netz–Blöcken werden über das Geister–Zell Verfahren durchgeführt. Dabei werden um die Netzzellen des Rechengebietes herum eine oder mehrere Zell–Schichten gelegt. Bei der Block–Kopplung werden die Zustände in den Geisterzellen mit den Zuständen am inneren Rand des angrenzenden Blocks befüllt. Die Zustände in den Geisterzellen werden dann für die Flussberechnung herangezogen, wobei je nach Randtyp unterschiedliche Flussfunktionen verwendet werden. An festen Wänden werden die Zustände in den Geisterzellen zur Erfüllung der gewünschten Randbedingung (reibungsbehaftet, rotierend oder stehend bzw. reibungsfrei) vom Inneren des

Gebiets nach außen extrapoliert. Um Netz–Zellen im Bereich der Wandgrenzschicht zu sparen, kann diese durch Wandfunktionen modelliert werden. Zur Ausnutzung von Rotationssymmetrie wird nur ein Segment des Ring–Raumes modelliert und an den entstehenden Rändern periodische Randbedingungen vorgegeben. Die stationäre Kopplung drehender und stehender Reihen geschieht entsprechend Dentons Mischungseben–Ansatz [106] durch Kopplung umfangsgemittelter Zustände. Der Strömungszustand wird am Eintritt des Rechengebiets durch Totaldruck, Totaltemperatur und Strömungsrichtung vorgegeben. Am Austritt wird eine Verteilung des statischen Gegendrucks spezifiziert. Alternativ kann ein statischer Druck bei mittlerer Kanalhöhe vorgegeben werden und es wird gefordert, dass die Differentialgleichung

$$dp = \rho U_\vartheta^2 \frac{dr}{r}, \tag{3.13}$$

erfüllt wird. Hierbei bezeichnet p den statischen Gegendruck und U_ϑ die Umfangskomponente der Strömungsgeschwindigkeit.

Sowohl Mischungs–Ebenen als auch Ein– und Austritt in das Gebiet nutzen nichtreflektierende Randbedingungen nach Giles [107], um unphysikalische Reflektionen von Wellen an den künstlichen Gebietsgrenzen zu unterdrücken.

3.5 Auswertung

Für die Auslegung von Turbomaschinenstufen existiert eine Vielzahl verwendeter Bewertungskriterien. Die wichtigste Klasse von Bewertungsgrößen stellen Bilanzen zwischen Ein- und Austrittsflächen eines Kontrollvolumens dar. Typische Bilanzkenngrößen sind dabei der Massenstrom, das Totaldruckverhältnis, Wirkungsgrade oder Verlustkenngrößen. Jedoch unterscheidet sich die Definition dieser Größen aufgrund der Art der simulierten Stufe (Verdichter, Turbine bzw. mit oder ohne Leitschaufeln versehener Strömungskanal) und den jeweiligen Auslegungsbedürfnissen. Auch müssen gelegentlich Einblasungen, Zapfstellen oder mehrere Eintritts- bzw. Austrittsebenen vorgesehen werden. Beispielsweise wenn bei Nebenstromtriebwerken die Aufteilung oder Zusammenführung der unterschiedlichen Stromwege teil des Kontrollvolumens ist. In manchen Fällen müssen radiale Verteilungen von umfangsgemittelten Bewertungsgrößen eingehalten werden, oder Kriterien auf der Schnittlinie der Schaufeloberfläche mit einer Stromfläche berücksichtigt werden. Daher wird die Berechnung von Zielfunktionen im vorliegenden Fall durch ein eigenes Programm realisiert, welches Mittelungs- und Verschneidungsroutinen sowie typische Auswertegrößen bereitstellt, jedoch modular um weitere Auswertungsberechnungen erweiterbar ist [108].

4 Adjungierte Simulation

Aus einem aerodynamischen Simulationsmodell können in der Regel Ausgabewerte, jedoch nicht deren Ableitung, gewonnen werden. Und obwohl in der Literatur eine Fülle gradientenfreier Optimierungsverfahren beschrieben wird [78], sind gradientenbasierte Optimierungsverfahren für lokale Optimierungsprobleme effizienter [80] und bereits etabliert (vgl. Kapitel 9 in [76]). Auch für globale Optimierungsmethoden können Gradienten die Effizienz bei der Bildung hochdimensionaler Ersatzmodelle steigern [70, 109]. Eine Voraussetzung hierfür ist jedoch, dass die Kosten der zusätzlichen Gradientenberechnung nicht die Beschleunigung zunichte machen. Bei der naheliegenden Art, Ableitungen mit finiten Differenzen zu berechnen, steigt die Zahl benötigter Auswertungen der teuren Bewertungsfunktion, linear mit der Anzahl freier Parameter. Für hochdimensionale Optimierungsprobleme ist diese Vorgehensweise daher ineffizient. In diesen Fällen kann das Adjungiertenverfahren eingesetzt werden.

4.1 Linearisierte und Adjungierte Berechnung von Gradienten

Das Adjungiertenverfahren berechnet Ableitungen von Auswertegrößen aus Simulationen so, dass die Berechnungskosten nicht wesentlich mit der Zahl freier Parameter steigen, sondern hauptsächlich von der Zahl der abzuleitenden Auswertegrößen abhängen.

Die wesentliche Idee kann wie folgt skizziert werden: Man möchte Ableitungen einer Funktion $\mathbf{y} = f(\mathbf{x})$ bestimmen wobei $\mathbf{x} \in \mathbb{R}^n$ und $\mathbf{y} \in \mathbb{R}^m$. In diesem Kontext heißt diese Funktion primal. Man kann sich unter x_j die n Entwurfsvariablen und unter f_i die m betrachteten Ausgabegrößen einer Simulation vorstellen. Für die hier betrachteten Optimierungsprobleme ist $n \gg m$. Für die Optimierung benötigt man alle partiellen Ableitungen von $\mathbf{f}$, also die Jacobi-Matrix $\frac{\partial f_i}{\partial x_j}$.

Es existieren zwei Arten die Einträge dieser Matrix zu berechnen: Spaltenweise per Vorwärts-Differentiation auch linearisiertes Verfahren genannt, oder zeilenweise durch Rückwärts-Differentiation, die auch adjungiertes Verfahren heißt. Beide berechnen mit einem Simulationslauf ein Produkt der Jacobi-Matrix $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ mit einem beliebigen reellen Vektor (Richtungsableitung). Das linearisierte Verfahren berechnet für eine vorgegebene Störung der $\dot{\mathbf{x}}$ Eingangsgrößen die linearisierte Auswirkung auf die Ausgabegrößen

$$\dot{\mathbf{y}} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \dot{\mathbf{x}}. \tag{4.1}$$

Dagegen wird beim adjungierten Verfahren ein Vektor aus dem Zielfunktionsraum $\mathbb{R}^m$,

mit der transponierten Jacobi-Matrix multipliziert[1]

$$\bar{\mathbf{x}} = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)^T \bar{\mathbf{y}}. \tag{4.2}$$

Die Produkte können algorithmisch ausgewertet werden, ohne die vollständige Jacobi-Matrix aufstellen zu müssen.

Wählt man als Ableitungsrichtung $\dot{\mathbf{x}}$ n-dimensionale Einheitsvektoren bzw. für $\bar{\mathbf{y}}$ m-dimensionale Einheitsvektoren erhält man mit einer Auswertung eine Spalte oder einer Zeile aus der Jacobi-Matrix:

$$\dot{\mathbf{y}} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_j} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots & & \vdots \\ \frac{\partial f_i}{\partial x_1} & \cdots & \frac{\partial f_i}{\partial x_j} & \cdots & \frac{\partial f_i}{\partial x_n} \\ \vdots & & \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_j} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix} \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \quad \bar{\mathbf{x}} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_j} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots & & \vdots \\ \frac{\partial f_i}{\partial x_1} & \cdots & \frac{\partial f_i}{\partial x_j} & \cdots & \frac{\partial f_i}{\partial x_n} \\ \vdots & & \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_j} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}^T \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}$$

Bei der linearisierten Vorgehensweise (links) erhält man in einem Durchgang die Ableitung aller Ausgangsgrößen nach einer Eingangsgröße. In der adjungierten Vorgehensweise (rechts) wird dagegen die Ableitung einer Ausgangsgröße nach allen Eingangsgrößen berechnet.

In den typischen Auslegungsoptimierungen betrachtet man mehr Entwurfsvariablen als Zielfunktionen und Nebenbedingungen. Die Jacobi-Matrix besitzt also mehr Spalten als Zeilen. In den folgenden Abschnitten wird gezeigt, dass eine linearisierte und eine adjungierte Auswertung etwa gleich aufwändig sind. Daher macht das Adjungiertenverfahren die Berechnung von Ableitungen effizienter, oder überhaupt erst erschwinglich wenn viel mehr Eingabe- als Ausgabegrößen betrachtet werden.

Für den Fall iterativer Lösungsmethoden existiert eine Methode, um einen effizienten adjungierten Löser zu implementieren. Im Folgenden werden zwei verbreitete Herleitungen dieser Methode dargestellt. Jede Herleitung bildet die Grundlage für eine der später dargestellten Implementierungen adjungierter Löser (vgl. Abs. 6.2 und Abs. 6.3). Die Herleitungen basieren auf unterschiedlichen Formulierungen der primalen Zustandsgleichung und führen daher auch zunächst zu unterschiedlichen Ergebnissen. Nach der Vorstellung der Herleitungen wird deren Äquivalenz gezeigt, sofern sie auf die gleiche Formulierung der primalen Zustandsgleichung angewandt werden.

4.2 Herleitung mittels des Satzes über implizite Funktionen

In der simulationsbasierten Auslegung berechnet man m verschiedene Zielfunktionswerte

$$y = J(q, x) \quad \text{mit } J : \mathbb{R}^l \times \mathbb{R}^n \to \mathbb{R}^m \tag{4.3}$$

[1] In der linearen Algebra ist adjungieren das komplexwertige äquivalent zum transponieren. Hiernach ist das Adjungiertenverfahren benannt.

die von den Netzkoordinaten $x \in \mathbb{R}^n$ und den l Zustandsgrößen q abhängt. Beispiele für Zielfunktionen sind Wirkungsgrad, Druckverhältnis oder Massenstrom einer Turbomaschinenkomponente. Das durchströmte Volumen wird durch das Rechennetz x diskretisiert und die zugehörige Strömungslösung heißt q. Diese Strömungslösung ist eine Funktion der Netzkoordinaten, definiert durch die implizite Funktion

$$w \equiv R(q, x) = 0 \quad \text{mit } R : \mathbb{R}^l \times \mathbb{R}^n \to \mathbb{R}^l. \tag{4.4}$$

Man kann sich darunter die Residuen der Erhaltungsgleichungen vorstellen, die für eine auskonvergierte Strömungslösung $q = q^*$ verschwinden.

Es wird vorausgesetzt, dass Gl. 4.4 dem Satz über implizite Funktionen genügt und ihre Jacobi–Matrix

$$\frac{\partial R}{\partial q} \in \mathbb{R}^{l \times l} \tag{4.5}$$

in einer Umgebung um die Lösung q^* nichtsingulär ist. Um die Jacobi–Matrix für Gl. 4.3 zu erhalten, leitet man $y = J(q(x), x)$ mittels Kettenregel bei $q = q^*$ ab

$$\frac{\mathrm{d}}{\mathrm{d}\,x} J(q(x), x) = \frac{\partial J}{\partial x} + \frac{\partial J}{\partial q} \frac{\mathrm{d}\,q}{\mathrm{d}\,x}. \tag{4.6}$$

Die Terme $\partial J/\partial x$ und $\partial J/\partial q$ können durch Differenzieren des Post-Processings berechnet werden. Man erhält $\mathrm{d}q/\mathrm{d}x$ in einer Umgebung von q^* durch Ableiten von Gl. 4.4:

$$\frac{\mathrm{d}}{\mathrm{d}\,x} 0 = \frac{\mathrm{d}}{\mathrm{d}\,x} R(x, q^*) \tag{4.7}$$

$$\Leftrightarrow 0 = \frac{\partial R}{\partial x} + \frac{\partial R}{\partial q} \frac{\mathrm{d}\,q}{\mathrm{d}\,x} \tag{4.8}$$

$$\Leftrightarrow \frac{\mathrm{d}\,q}{\mathrm{d}\,x} = -\left(\frac{\partial R}{\partial q}\right)^{-1} \left(\frac{\partial R}{\partial x}\right) \tag{4.9}$$

Durch Einsetzen in Gl. 4.6 ergibt sich:

$$\frac{\mathrm{d}\,J}{\mathrm{d}\,x} = \frac{\partial J}{\partial x} - \frac{\partial J}{\partial q} \left(\frac{\partial R}{\partial q}\right)^{-1} \frac{\partial R}{\partial x}. \tag{4.10}$$

Das Produkt der drei Matrizen auf der rechten Seite kann man in zwei verschiedenen Reihenfolgen auswerten: Linearisiert

$$\frac{\mathrm{d}\,J}{\mathrm{d}\,x} = \frac{\partial J}{\partial x} - \frac{\partial J}{\partial q} \underbrace{\left[\left(\frac{\partial R}{\partial q}\right)^{-1} \frac{\partial R}{\partial x}\right]}_{\frac{\partial q}{\partial x} \in \mathbb{R}^l \times \mathbb{R}^n} \tag{4.11}$$

oder adjungiert

$$\frac{\mathrm{d}\,J}{\mathrm{d}\,x} = \frac{\partial J}{\partial x} - \underbrace{\left[\frac{\partial J}{\partial q} \left(\frac{\partial R}{\partial q}\right)^{-1}\right]}_{\frac{\partial J}{\partial w} \in \mathbb{R}^m \times \mathbb{R}^l} \frac{\partial R}{\partial x}. \tag{4.12}$$

Zur Berechnung dieser Terme formuliert man zu den geklammerten Ausdrücken die äquivalenten Gleichungssysteme, löst diese für $\partial q/\partial x$ bzw. $\partial J/\partial w$ und multipliziert anschließend mit dem verbleibenden Term, also $\partial J/\partial q$ oder $\partial R/\partial x$.

Für die linearisierte Vorgehensweise formuliert man daher aus dem geklammerten Term in Gl. 4.11 ein Gleichungssystem:

$$\frac{\mathrm{d}\,q}{\mathrm{d}\,x} = -\left(\frac{\partial R}{\partial q}\right)^{-1} \frac{\partial R}{\partial x} \tag{4.13}$$

$$\Leftrightarrow \frac{\partial R}{\partial q}\frac{\mathrm{d}\,q}{\mathrm{d}\,x} = -\frac{\partial R}{\partial x} \tag{4.14}$$

Der unbekannte Term $\mathrm{d}q/\mathrm{d}x$ soll berechnet werden, jedoch steht auf der linken Seite der Gleichung noch ein Produkt zweier Matrizen. Um zu Gleichungssystemen der Form $Ax = b$ zu gelangen, stellt man für jede Spalte von $\mathrm{d}q/\mathrm{d}x$ ein eigenes Gleichungssystem

$$\frac{\partial R}{\partial q}\frac{\mathrm{d}\,q}{\mathrm{d}\,x_j} = \frac{\partial R}{\partial x_j} \quad j = 1..n \tag{4.15}$$

auf. Man erkennt, dass diese Gleichung unabhängig von der Wahl des Zielfunktionals i ist, aber für jeden Enwurfsparameter x_j getrennt gelöst werden muss.

Nach Bestimmung von $\mathrm{d}q/\mathrm{d}x$, erhält man die ursprünglich gewünschten Ableitungen durch Auswertung von

$$\frac{\mathrm{d}\,J_i}{\mathrm{d}\,x_j} = \frac{\partial J_i}{\partial x_j} + \frac{\partial J_i}{\partial q}\frac{\mathrm{d}\,q}{\mathrm{d}\,x_j} \quad i = 1..m, j = 1..n. \tag{4.16}$$

Für die adjungierte Vorgehensweise wählt man die zweite Möglichkeit zur Klammerung von Gl. 4.12. Da $J(q, x)$ über q eine implizite Funktion von $w \equiv R(q, x) = 0$ ist, schreibt man

$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial q}\left(\frac{\partial R}{\partial q}\right)^{-1} \tag{4.17}$$

und

$$\frac{\partial J}{\partial w}\frac{\partial R}{\partial q} = \frac{\partial J}{\partial q} \tag{4.18}$$

Durch Transponierung und Festlegung auf einzelne Zielfunktionen J_i und der Definition des Vektors adjungierter Variablen

$$\Psi_i = \left(\frac{\partial J_i}{\partial w}\right)^T \tag{4.19}$$

erhält man die adjungierten Gleichungssysteme:

$$\left(\frac{\partial R}{\partial q}\right)^T \Psi_i = \left(\frac{\partial J_i}{\partial q}\right)^T \quad i = 1..m \tag{4.20}$$

Man erkennt, dass diese nicht von der Wahl eines Entwurfsparameters j abhängen, jedoch für jedes J_i gelöst werden müssen. Durch Lösen erhält man Ψ_i, und damit die gewünschten Ableitungen durch die Auswertung von

$$\frac{\mathrm{d}\, J_i}{\mathrm{d}\, x_j} = \frac{\partial J_i}{\partial x_j} - \Psi_i^T \left(\frac{\partial R}{\partial x_j} \right) \quad i = 1..m, j = 1..n. \tag{4.21}$$

In beiden Fällen (4.15, 4.20) müssen zur Berechnung der vollständigen Jacoi-Matrix mehrere lineare Gleichungssysteme gelöst werden. Der Aufwand der darauf folgenden Auswertungen (4.16, 4.21) ist dagegen vernachlässigbar, da es sich lediglich um Skalarprodukte und Vektor-Additionen handelt. Auch wenn die beteiligten Skalarprodukte sehr groß sind, ist der Aufwand vernachlässigbar, da ihre Komplexität mit $\mathcal{O}(n)$ unbedeutend gegenüber des Aufwands für die Lösung der linearen Gleichungssysteme ist.

Die dabei verwendeten System–Matrizen in Gl. 4.15 und Gl. 4.20 sind bis auf Transposition gleich, haben somit auch gleiche Spektren und gleiche Konditionszahlen weshalb das Lösen der Gleichungssysteme auch gleich aufwändig ist.

Es müssen im Fall der linearen Vorgehensweise n Gleichungssysteme und im adjungierten Fall m Gleichungssysteme gelöst werden. Ist also die Zahl freier Parameter n viel größer als die Zahl betrachteter Zielfunktionen m, ist die adjungierte Methode für die Berechnung der Jacobi-Matrix $\mathrm{d}J/\mathrm{d}x$ effizienter.

4.3 Herleitung mittels Langrange-Formalisumus

Im vorherigen Abschnitt wurden das linearisierte Verfahren und adjungierte Verfahren hergeleitet. In diesem Abschnitt wird eine alternative Herleitungsform des adjungierten Verfahrens dargestellt, welche in der Literatur verbreitet ist und die Grundlage der in Abschnitt 6.3 diskutierten Implementierung ist. Da die Rechenzeitersparnis des Adjungiertenverfahrens bereits im vorhergehenden Abschnitt erläutert wurde, wird hier, der Einfachheit halber, nur ein einzelnes aerodynamisches Funktional angenommen

$$J(q_N, x) \quad \text{mit } J : \mathbb{R}^l \times \mathbb{R}^n \to \mathbb{R}, \tag{4.22}$$

an, welches auf dem Ergebnis des Strömungslösers q_N nach der N-ten Iteration basiert. Dieses ist definiert durch

$$G(q_{i-1}, x) = q_i \quad i = 1, \ldots, N. \tag{4.23}$$

Unter G kann man sich das gesamte Lösungsverfahren vorstellen, welche aus einem gegebenen Strömungszustand eine neue Näherung der Lösung produziert. Im Unterschied zum vorherigen Abschnitt werden also nicht nur die Residuen R einbezogen, sondern auch das Verfahren welches einen neuen Strömungszustand daraus berechnet.

Zur Berechnung von $\mathrm{d}J/\mathrm{d}x$ formuliert man die entsprechende Lagrange-Funktion

$$L(Q, x, \Lambda) = J(q_N, x) + \sum_{i=1}^{N} \lambda_i^T (G(q_{i-1}, x) - q_i), \tag{4.24}$$

wobei $Q = (q_i)$ die Zustände, und $\Lambda = (\lambda_i)$ die Lagrange-Multiplikatoren aller Iterationen $i = 1..N$ bezeichnet. Genügt Q der Voraussetzung in Gl. 4.23 ist

$$\sum_{i=0}^{N}(G_i(q_{i-1}, x) - q_i) = 0 \tag{4.25}$$

und daher

$$\frac{\mathrm{d}\, J(q_N, x)}{\mathrm{d}\, x} = \frac{\mathrm{d}\, L(Q(x), x, \Lambda)}{\mathrm{d}\, x}. \tag{4.26}$$

Die totale Ableitung von L nach den Kontrollvariablen x ist

$$\frac{\mathrm{d}\, L(Q(x), x, \Lambda)}{\mathrm{d}\, x} = \frac{\partial L}{\partial Q}\frac{\mathrm{d}\, Q}{\mathrm{d}\, x} + \frac{\partial L}{\partial x}. \tag{4.27}$$

Der entscheidende Kunstgriff ist, dass bei Erfüllung von Gl. 4.23 die λ in Gl. 4.24 beliebig gewählt werden können.

Man bestimmt Λ, so dass $\partial L(Q(x), x, \Lambda)/\partial Q \stackrel{!}{=} 0$.

Da

$$\frac{\partial L}{\partial Q}\frac{\mathrm{d}\, Q}{\mathrm{d}\, x} = \sum_{i=1}^{N}\frac{\partial L}{\partial q_i}\frac{\mathrm{d}\, q_i}{\mathrm{d}\, x} \tag{4.28}$$

müssen

$$\begin{aligned}\frac{\partial L}{\partial q_N} &= \frac{\partial J(q_N, x)}{\partial q_N} - \lambda_N \stackrel{!}{=} 0\\ \frac{\partial L}{\partial q_j} &= \lambda_{j+1}\frac{\partial G(q_j, x)}{\partial q_j} - \lambda_j \stackrel{!}{=} 0\end{aligned} \tag{4.29}$$

sein. Hieraus erhält man eine Berechnungsvorschrift für λ_j die für $j = N..1$ durchlaufen werden kann

$$\begin{aligned}\lambda_N^T &= \frac{\partial J(q_N, x)}{\partial q}\\ \lambda_i^T &= \lambda_{i+1}^T\frac{\partial}{\partial q}G(q_{i-1}, x).\end{aligned} \tag{4.30}$$

Mit diesen Werten für λ_j ist

$$\frac{\mathrm{d}\, J}{\mathrm{d}\, x} = \frac{\mathrm{d}\, L(Q(x), x, \Lambda)}{\mathrm{d}\, x} = \frac{\partial J(q_N(x), x)}{\partial x} + \sum_{i=1}^{N}\lambda_i^T\frac{\partial G(q_{i-1}, x)}{\partial x} \tag{4.31}$$

eine Berechnungsvorschrift für die gewünschten Ableitungen des Zielfunktionals.

Die Vorschriften in Glg. 4.30, 4.31 sind äquivalent zum Resultat, das man durch Anwendung von algorithmischer Differentiation (s. Abschnitt 5.4) auf den kompletten Programmcode eines Strömungslösers erhalten würde. Daher heißt dieses Verfahren Black–Box adjungierte. Man sieht in Gl. 4.30, dass sämtliche Strömungszuständes in umgekehrter Reihenfolge benötigt werden, um die adjungierten Zustände λ_i zu berechnen. Diese

müssen daher im Vorwärtslauf gespeichert werden. Aufgrund ihres hohen Ressourcenbedarfs ist diese Methode weniger geeignet Optimierungen zu beschleunigen. Sie dient jedoch als Validierungsbasis für eine effizientere Berechnungsvorschrift, welche sich die Fixpunkteigenschaft für G zunutze macht.

Um eine effizientere Berechnungsvorschrift zu erhalten nimmt man an, dass die Iterationsvorschrift in einen Fixpunkt $G(q^*, x) = q^*$ konvergiert. An diesem Fixpunkt möchte man die Ableitung des Zielfunktionals nach den Kontrollparametern

$$\frac{\mathrm{d}\, J(q^*(x), x)}{\mathrm{d}\, x} \tag{4.32}$$

berechnen. Die zugehörige Lagrange-Funktion ist

$$L(q^*(x), x, \mu) = J(q^*(x), x) + \mu^T (G(q^*, x) - q^*). \tag{4.33}$$

Aus

$$\frac{\partial L(q^*, x, \mu)}{\partial q} = \frac{\partial J(q^*, x)}{\partial q} + \mu^T \left(\frac{\partial G(q^*, x)}{\partial q} - I \right) = 0, \tag{4.34}$$

mit der Einheitsmatrix I, erhält man eine Fixpunktgleichung für μ:

$$\mu^T = \frac{\partial J(q^*, x)}{\partial q} + \mu^T \frac{\partial G(q^*, x)}{\partial q}, \tag{4.35}$$

welche sich in folgende Iterationsvorschrift umwandeln lässt:

$$\begin{aligned} \mu_1^T &= \frac{\partial J(q^*, x)}{\partial q} \\ \mu_{i+1}^T &= \frac{\partial J(q^*, x)}{\partial q} + \mu_i^T \frac{\partial G(q^*, x)}{\partial q}. \end{aligned} \tag{4.36}$$

Eine frühe Beschreibung dieses Verfahrens im Kontext automatischer Differentiation liefert Christianson [110]. Ebendort wird dargelegt, dass ein solches Iterationsschema $\mu_{i+1} = \bar{G}(q^*, x, \mu_i)$ für einen kontraktiven Fixpunkt q^* von G, ebenfalls einen Fixpunkt μ^* besitzt und zu diesem asymptotisch die gleiche Konvergenzrate aufweist wie G bei q^*. Diese Iterationsvorschrift nennt Christianson „fixpoint reverse recurrence“, zu deutsch Fixpunkt-Rückwärts-Iteration. Aus einer sehr starken Verkürzung des Titels der Veröffentlichung [110] hat sich hierfür auch der Name „reverse-accumulation“ etabliert, wobei dieser Namensteil eigentlich nur den Rückwärtsmodus algorithmischer Differentiation bezeichnet.

Mit einem konvergierten μ^* kann man die gewünschten Ableitungen des Zielfunktionals berechnen

$$\frac{\mathrm{d}\, J}{\mathrm{d}\, x} = \frac{\partial}{\partial x} L(q^*(x), x, \mu^*) = \frac{\partial}{\partial x} J(q^*(x), x) + \mu^{*T} \frac{\partial}{\partial x} G(q^*, x) \tag{4.37}$$

An einem Fixpunkt $q_i = q^*$ durchgeführt, sind die Black-Box-Iteration Gl. 4.31 und die Fixpunkt-Iteration 4.37 äquivalent. Die Black-Box-Iteration (Gl. 4.30) ist dann im i-ten Schritt

$$\lambda_{N-i-1}^T = \frac{\partial J}{\partial q} \left(\frac{\partial G}{\partial q} \right)^{i-1}. \tag{4.38}$$

Eingesetzt in die Black-Box Auswertung Gl. 4.31 ist

$$\frac{\mathrm{d}\,J}{\mathrm{d}\,x} = \frac{\partial J}{\partial x} + N\frac{\partial J}{\partial q}\sum_{i=1}^{N}\left(\frac{\partial G}{\partial q}\right)^{i-1}\frac{\partial G}{\partial x}. \tag{4.39}$$

Die Fixpunkt-Iteration Gl. 4.36 ist nach N Schritten

$$\begin{aligned}\mu_N^T &= \frac{\partial J}{\partial q} + \left(\frac{\partial J}{\partial q} + \left(\frac{\partial J}{\partial q}\cdots\frac{\partial J}{\partial q}\frac{\partial G}{\partial q}\right)\frac{\partial G}{\partial q}\right)\frac{\partial G}{\partial q} \\ &= \sum_{i=1}^{N}\frac{\partial J}{\partial q}\left(\frac{\partial G}{\partial q}\right)^{i-1} \\ &= N\frac{\partial J}{\partial q}\sum_{i=1}^{N}\left(\frac{\partial G}{\partial q}\right)^{i-1}.\end{aligned} \tag{4.40}$$

Eingesetzt in die Fixpunkt-Auswertung Gl. 4.37 erhält man das gleiche Ergebnis wie in Gleichung 4.39.

4.4 Zusammenhang der Herleitungen

In den vorangegangenen Abschnitten wurden zwei verschiedene Herleitungen des Adjungiertenverfahrens für iterative Löser vorgestellt. Sie wurden jedoch auf unterschiedliche Formulierungen der Definition des Lösers angewandt. Die Herleitung über den Satz impliziter Funktionen basierte auf der Annahme $R(q^*, x) = 0$, während die Bestimmung der Lagrange-Multiplikatoren für die Fixpunkt-Form der gesamten Iterationsvorschrift des Strömungslösers $q^* = G(q^*, x)$ formuliert wurde. Das liegt daran, dass jede Herleitung Grundlage für eine der beiden Implementierung ist, welche in den Abschnitten 6.2 und 6.3 vorgestellt werden.

Im Folgenden soll gezeigt werden, dass beide Herleitungen zu gleichen Ergebnissen führen, wenn man Sie auf die gleiche Löserdefinition anwendet. Der Zusammenhang der Löserdefinition kann als

$$G(q, x) = q - P(q, x)R(q, x) \tag{4.41}$$

geschrieben werden, wobei $P(q, x)$ einen Präkonditionierer des Iterationsverfahrens darstellt.

Wendet man die Herleitungsvorschrift des Satzes über implizite Funktionen auf die Fixpunkt-Darstellung an, so erhält man bei einen Ausdruck für $\mathrm{d}q/\mathrm{d}x$, wobei die Ableitungen bei $q = q^*$ vorgenommen werden

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\,x}G(q, x) &= \frac{\mathrm{d}}{\mathrm{d}\,x}q, \\ \Leftrightarrow \frac{\partial G}{\partial q}\frac{\mathrm{d}\,q}{\mathrm{d}\,x} + \frac{\partial G}{\partial x} &= \frac{\mathrm{d}\,q}{\mathrm{d}\,x} \\ \Leftrightarrow \left(\frac{\partial G}{\partial q} - I\right)\frac{\mathrm{d}\,q}{\mathrm{d}\,x} &= -\frac{\partial G}{\partial x}\end{aligned} \tag{4.42}$$

$$\frac{\mathrm{d}\,q}{\mathrm{d}\,x} = -\left(\frac{\partial G}{\partial q} - I\right)^{-1} \frac{\partial G}{\partial x}. \tag{4.43}$$

Diesen Ausdruck setzt man in die zu berechnende Ableitung

$$\frac{\mathrm{d}\,J}{\mathrm{d}\,x} = \frac{\partial J}{\partial q}\frac{\mathrm{d}\,q}{\mathrm{d}\,x} + \frac{\partial J}{\partial x} \tag{4.44}$$

ein:

$$\frac{\mathrm{d}\,J}{\mathrm{d}\,x} = \underbrace{\left[-\frac{\partial J}{\partial q}\left(\frac{\partial G}{\partial q} - 1\right)^{-I}\right]}_{\frac{\partial J}{\partial w}} \frac{\partial G}{\partial x} + \frac{\partial J}{\partial x}. \tag{4.45}$$

Der Ausdruck in den eckigen Klammern erfüllt

$$\begin{aligned} \frac{\partial J}{\partial w} &= -\frac{\partial J}{\partial q}\left(\frac{\partial G}{\partial q} - I\right)^{-1} \\ \Leftrightarrow \frac{\partial J}{\partial w}\left(\frac{\partial G}{\partial q} - I\right) &= -\frac{\partial J}{\partial q} \end{aligned} \tag{4.46}$$

und lässt sich mit $\mu^T = \frac{\partial J}{\partial w}$ in Fixpunktform schreiben:

$$\mu^T = \frac{\partial J}{\partial q} + \mu^T \frac{\partial G}{\partial q} \tag{4.47}$$

Daraus lässt sich die Iterationsvorschrift

$$\mu_{i+1}^T = \frac{\partial J}{\partial q} + \mu_i^T \frac{\partial G}{\partial q} \tag{4.48}$$

bilden. Dies entspricht der Iterationsvorschrift in Gl. 4.36.

5 Berechnung partieller Ableitungen numerischer Prozeduren

Für die Implementierung eines linearisierten oder adjungierten Verfahrens müssen die auftretenden partiellen Ableitungen berechnet werden. Da die zu differenzierenden Funktionen als Teile des Strömungslösers, implementiert sind, müssen also numerische Prozeduren differenziert werden. Für die Auswertung partieller Ableitungen einer Prozedur werden in der Literatur vier verschiedene Techniken beschrieben, die im Folgenden wiedergegeben werden.

Der Begriff Prozedur wird hier in Abgrenzung zur Funktion wie folgt verstanden: Eine Funktion berechnet ihr Ergebnis ausschließlich aus ihren Argumenten und verändert ansonsten nicht den Zustand des Programmspeichers. Das heißt Funktionen sind frei von Seiteneffekten: Eingabe- und Ausgabe sind bei Funktionen klar definiert. Prozeduren hingegen können auf Variablen zugreifen, die nicht in der Argumentliste auftauchen (globale Variablen) und auch den Zustand des Speichers verändern. Funktionen nach obiger Definition gibt es nur in echten funktionalen Programmiersprachen. Die hier betrachteten Sprachen sind prozedurale Programmiersprachen. Darin gibt es nur Prozeduren, auch wenn diese im Kontext der Programmiersprache gelegentlich anders genannt werden.

5.1 Manuelles Differenzieren

Die naheliegendste Möglichkeit, die Ableitung einer Prozedur f zu berechnen, ist es, die Ableitung eigenhändig herzuleiten und ihre Berechnung in einer differenzierten Routine zu implementieren. Für die Berechnung der Ableitung brauchen wir, außer in trivialen Fällen, zunächst die Formelschreibweise der Berechnungen in f. Manuelles Differenzieren einer Prozedur besteht also aus drei Schritten:

1. Beschreibung der Prozedur als mathematische Formel
2. Differenzieren der Formel
3. Implementierung der differenzierten Berechnung in einer neuen Prozedur

In Schritt 2 kann man sowohl die Vorwärtsdifferentiation als auch die Rückwärtsdifferentiation anwenden, also die Vorwärtsableitung

$$\dot{y} = \frac{\partial f}{\partial x}\dot{x} \tag{5.1}$$

oder die adjungierte Ableitung

$$\bar{x} = \left(\frac{\partial f}{\partial x}\right)^T \bar{y} \tag{5.2}$$

berechnen. Für viele dabei auftretende Standardprobleme gibt es fertige Rezepte in der Literatur [111, 112]. Die Differentiation in Schritt 2 kann der Entwickler selbst durchführen, oder ein Computer-Algebra-Systems (CAS) benutzen. Einige Computer-Algebra-Systeme sind auch in der Lage, die gewonnenen Gleichungen direkt in Quelltext zu übersetzen, wodurch Schritt 3 erheblich erleichtert wird.

Dabei wird auch eine bedeutende Fehlerquelle dieser Vorgehensweise vermieden: Inkonsistenzen zwischen differenzierter Formel und ihrer Implementierung. Allerdings ist dieser automatisch generierte Code in der Regel für Menschen schwer zu verstehen.

Die Formulierung als mathematische Gleichungen ermöglicht Umformungen und Vereinfachungen, die bei reiner Betrachtung der in Code implementierten Berechnungen möglicherweise nicht sichtbar geworden wären. Dies liegt daran, dass die mathematische Gleichungsform eine Abstraktion von der konkreten Implementierung darstellt und in ihr der gesamte Apparat algebraischer Umformungen zur Verfügung steht. Der Code, der bei dieser Vorgehensweise in der Implementierung entsteht, kann daher effizienter sein als bei den anderen Vorgehensweisen, welche eine Abstraktion der Berechnungen nicht erzwingen.

An dieser Stelle sind zwei Nachteile der analytischen Vorgehensweise bedeutsam: Zum einen machen Menschen Fehler, die hier bei der Extraktion der Formelschreibweise, bei ihrer Differentiation, bei der Umformung sowie bei der Implementierung auftreten können, und dazu führen, dass die implementierten differenzierten Routinen nicht der differenzierten Funktionsberechnung entsprechen. Ein weiteres Problem betrifft Programme die aktiv fortentwickelt werden. Wird eine primale Routine geändert, zu welcher bereits eine differenzierte Variante implementiert wurde, muss die differenzierte Variante aktualisiert werden. Andernfalls werden primale und differenzierte Routinen inkonsistent.

Dieses Problem des analytischen Verfahrens tritt um so häufiger auf, je mehr Entwickler gleichzeitig beteiligt sind. Typischerweise ist der Großteil der Entwickler dann mit dem primalen Programm und ein wesentlich kleinerer Teil mit der adjungierten Variante betraut. Ein Entwickler des primalen Lösers weiß daher nicht, ob eine Änderung die er vornimmt möglicherweise eine differenzierte Routine inkonsistent macht. Eine Möglichkeit dies zu Erkennen wären automatische Tests der differenzierten Routinen (auch als Unit-Tests bezeichnet). Diese sind jedoch mit Unsicherheiten behaftet, da die naheliegenden Vergleichswerkzeuge finite Differenzen sind. Im Folgenden werden wir sehen, dass diese von numerischen Ungenauigkeiten betroffen sind.

5.2 Finite Differenzen

Finite Differenzen werden häufig mit erster Ordnung Genauigkeit angewandt. Entweder als Vorwärtsdifferenzen

$$f'(x_0) = \frac{f(x_0 + \epsilon) - f(x_0)}{\epsilon} + \mathcal{O}(\epsilon) \tag{5.3}$$

oder Rückwärtsdifferenzen

$$f'(x_0) = \frac{f(x_0) - f(x_0 - \epsilon)}{\epsilon} + \mathcal{O}(\epsilon) \tag{5.4}$$

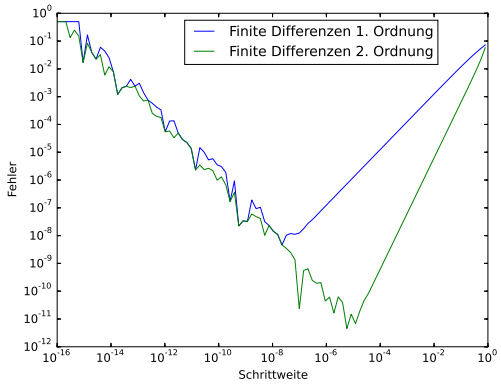


Abbildung 5.1: Abweichung finiter Differenzen von der analytischen Ableitung bezüglich des Schrittweitenparameters für die Funktion $1 - \sqrt{x}$

Aus dem arithmetischen Mittel der beiden Formulierungen lässt sich die zentrale Differenz mit zweiter Ordnung Genauigkeit darstellen:

$$f'(x_0) = \frac{f(x_0 + \epsilon) - f(x_0 - \epsilon)}{2\epsilon} + \mathcal{O}(\epsilon^2) \tag{5.5}$$

Während der Approximationsfehler für Vorwärts- und Rückwärtsdifferenz linear vom Schrittweitenparameter ϵ abhängt, sinkt bei der zentralen Differenz der Fehler quadratisch mit sinkendem ϵ. Jedoch kann ϵ nicht beliebig klein gewählt werden, da die Differenz im Zähler bei Berechnung in Gleitkommaarithmetik zu Auslöschungsfehlern führt [113]. Der optimale Kompromiss für die Schrittweite hängt vom Verlauf der Funktion f am Punkt x_0 ab. In Abbildung 5.1 ist der Verlauf des Fehlers finiter Differenzen für die Funktion $f(x) = 1 - \sqrt{x}$ für $x_0 = 1$ im Vergleich zur analytischen Ableitung dargestellt. Man sieht, dass die Verkleinerung der Schrittweite nur bis zu einer Untergrenze hin, den Approximationsfehler verkleinert. Danach führt eine Verringerung der Schrittweite zu einer Vergößerung des Fehlers durch Auslöschungseffekte. Die finiten Differenzen zweiter Ordnung erreichen eine höhere Genauigkeit schon bei größerer Schrittweite. Der Verlauf des Fehlers, insbesondere der Punkt mit dem niedrigstmöglichen Fehler ist abhängig von der zu differenzierenden Funktion und der Stelle an der differenziert werden soll. Finite Differenzen von Funktionen aus der Praxis sind typischerweise erheblich ungenauer als dieses analytische Beispiel. Es müsste für jede zu differenzierende Funktion und jede Stelle getrennt ein optimales ϵ gefunden werden, ohne dass eine exakte Vergleichslösung zur Verfügung steht. Daher behilft man sich in der Regel mit heuristischen Verfahren zur Schrittweitenfindung.

Zur Kostenbetrachtung finiter Differenzen nehmen wir Prozeduren mit n Gleitkommawerten als Eingabe und m Gleitkommawerten als Ausgabe an. Dann ist der Differenzenquotient zur Approximation einer Richtungsableitung mit dem Richtungsvektor $v \in \mathbb{R}^n$

$$D_v f(x) = \lim_{\epsilon \to 0} \frac{f(x + \epsilon \cdot v) - f(x)}{\epsilon}. \tag{5.6}$$

Diese Vorgehensweise entspricht der Vorwärtsdifferentiation $\dot{y} = \frac{\partial f}{\partial x} v$. Zur Approximation der Jacobi-Matrix $J(x_0) = \left(\frac{\partial f_i}{\partial x_j}\right)_{i=1..m,\, j=1..n}$ einer Prozedur werden mit finiten Differenzen erster Ordnung $n+1$ Prozedurausführungen gebraucht, im Fall finiter Differenzen zweiter Ordnung sind es $2n$ Ausführungen. Der Berechnungsaufwand für finite Differenzen ist somit linear von der Anzahl betrachteter Eingabegrößen abhängig. Ist die Auswertung der Funktion f teuer und es liegen viele freie Parameter vor, sind finite Differenzen daher ungünstig.

Finite Differenzen haben jedoch den Vorteil, dass man mit ihnen Prozeduren oder ganze Programme ohne Kenntnis der Implementierung alleine durch Auswertung für verschiedene Parameter differenzieren kann (Black-Box Differentiation). Aufgrund ihrer einfachen Anwendung werden Sie auch zur Validierung anderer Differentiationstechniken herangezogen.

5.3 Differenzenquotienten mit imaginärer Schrittweite

Auch wenn Differenzenquotienten mit imaginärer Schrittweite in der vorliegenden Arbeit keine Anwendung finden, sollen sie der Vollständigkeit halber dennoch vorgestellt werden. Differenzenquotienten mit imaginärer Schrittweite lösen das Auslöschungsproblem der soeben beschriebenen finiten Differenzen. Das Verfahren besteht in der Fortsetzung der zu differenzierenden Funktion in die komplexe Ebene und der Anwendung der Cauchy-Riemann'schen Differentialgleichungen. Dabei setzt man voraus, dass die zu differenzierende Funktion f reellwertig und ihre komplexe Fortsetzung holomorph ist, d.h. sie muss die Cauchy-Riemann'schen Differentialgleichungen erfüllen. Wir erhalten die Approximation der Ableitung

$$\frac{\partial f}{\partial x} \approx \frac{\mathrm{Im}[f(x+ih)]}{h}, \tag{5.7}$$

welche keine Differenz beeinhaltet und somit unanfällig für die numerische Auslöschungsproblematik der reellwertigen finiten Differenzen Approximation ist. Daher kann die Schrittweite h sehr klein gewählt werden, um eine sehr genaue Approximation der Ableitung zu erhalten.

Auch wenn dieser Ansatz in der Praxis Verwendung findet [114], so ist die Voraussetzung der komplexen Differenzierbarkeit von f eine sehr viel stärkere Forderung als die reelle, stückweise Differenzierbarkeit, welche für die algorithmische Differentitation gefordert wird.

In Ergebnisgenauigkeit, sowie Laufzeit– und Speicheranforderungen entspricht sie weitgehend [115, 12] der algorithmischen Differentiation im Vorwärtsmodus, deren Eigenschaften bezüglich nicht überall stetig differenzierbarer Funktionen eingehend untersucht ist (s. Kapitel 14 in [17]). Da alle beteiligten Berechnungen in die komplexe Ebene übertragen werden müssen, kann sie nur für Prozeduren angewandt werden, deren Quelltext man verändern kann und dessen Programmiersprache komplexe Arithmetik erlaubt.

Finite Differenzen komplexer Schrittweite können jedoch nützlich sein, um auf einfache

Weise exakte Validierungsergebnisse für andere Verfahren zu erhalten. In dieser Arbeit kommt für diese Zwecke algorithmisches Differenzieren im Vorwärtsmodus zum Einsatz.

5.4 Algorithmisches Differenzieren

Algorithmische Differentiation (AD) kann für Prozeduren, die als Quellcode vorliegen, auf algorithmische Weise, also durch ein Computerprogramm, eine differenzierte Prozedur generieren. Der Begriff automatische Differentiation ist ebenfalls verbreitet und bezeichnet dasselbe, jedoch führt der Begriff leicht zu überhöhten Erwartungen und wird daher hier vermieden. Die Vorbereitung eines Programms auf algorithmische Differentiation ist ein manueller, möglicherweise werkzeugunterstützter Prozess. Die darauf folgende, Berechnung von Ableitungen geschieht dann automatisch.

Die folgenden Abschnitte sind eine kurze Einführung in die Thematik, die sich an den Lehrbüchern [17] sowie [18] orientiert. Sie dienen vor allem dazu, die Begriffe und Grundeigenschaften der AD zusammenzufassen und eine Vorstellung zu vermitteln, wie AD funktioniert und angewendet werden kann. Für eine theoretisch fundierte Einführung und die tatsächliche Vorgehensweise zur Implementierung eines AD-Werkzeugs sollte der Leser auf die zuletzt genannten Bücher in der angegebenen Reihenfoge zurückgreifen.

Für die Differentiation einer Prozedur mittels AD setzt man Folgendes voraus:

- Ein- und Ausgabegrößen der Prozedur sind bekannt
- Die Prozedur besteht aus einer Folge elementarer Operationen auf Gleitkommazahlen deren Ableitungen bekannt sind
- Die Prozedur ist an der ausgewerteten Stelle mindestens einmal stetig differenzierbar
- Die Prozedur terminiert

Eine solche Prozedur kann man dann immer als eine Folge von Anweisungen betrachten, welche jeweils nur eine elementare Operation (z.B. $\varphi \in \{+, -, *, /, a^b, \exp, \sin, ...\}$) auf bis zu zwei Variablen durchführt und das Ergebnis einer neuen Variablen zuweist

$$v_j = \varphi_j(v_i)_{i \prec j} \quad \text{mit} \quad j = 0, ..., n+p+m-1 \quad \text{und} \quad n, p, m \in \mathbb{N}. \tag{5.8}$$

$$x = [v_j] \quad \text{mit} \quad j = 0, ..., n-1 \quad \text{Eingabevariablen} \tag{5.9}$$

$$v_{n+j} \quad \text{mit} \quad j = 0, ..., p-1 \quad \text{Zwischenergebnisse} \tag{5.10}$$

$$y = [v_{n+p+j}] \quad \text{mit} \quad j = 0, ..., m-1 \quad \text{Ausgabegrößen} \tag{5.11}$$

Die Notation $i \prec j$ bedeutet, dass Variable v_j unmittelbar von der Variable v_i abhängt. Weiterhin steht $(v_i)_{i \prec j}$ für die Liste von Variablen für die $i \prec j$ gilt, also alle Variablen die in der j-ten Berechnung vorkommen. Die Ableitung der Prozedur kann dann mittels Kettenregel aus den Ableitungen der Elementaroperationen berechnet werden.

Reale numerische Programme liegen nicht in dieser Darstellung vor, aber man kann sie, für einen gegebenen Eingabedatensatz, in eine solche Form transformieren. Zum Verständnis ist es lediglich wichtig, dass eine solche Transformation möglich ist. In der praktischen Umsetzung des algorithmischen Differenzierens müssen diese Transformationen nicht zwingend angewendet werden. Vor allem ist wichtig, dass diese Form nur für einen gegebenen Eingabedatensatz durchgeführt werden muss. Für diesen kann beispielsweise entschieden werden, welche Zweige einer Bedingung genommen werden, wie oft eine Schleife durchlaufen wird, welche Unterprozeduren aufgerufen werden.

Diese Notation ist äquivalent zu einem gerichteten zyklenfreien Graphen $G = (V, E)$ dessen Knoten ganze Zahlen $V \in \{0, ..., n + p + m - 1\}$ sind, die durch Kanten $E = \{(i, j) | i \prec j\}$ mit den Knoten verbunden sind von deren Ergebnissen sie unmittelbar abhängen.

Ein Beispiel soll dies veranschaulichen: Wir nehmen eine Prozedur an, die folgende Funktion implementiert

$$y = f(x_1, x_2, x_3) = \sin(x_1 \cdot x_2) + \cos({x_3}^2) \tag{5.12}$$

Diese lässt sich wie folgt zerlegen:

$$\begin{aligned} v_0 &= x_1 \\ v_1 &= x_2 \\ v_2 &= x_3 \\ v_3 &= v_0 \cdot v_1 \\ v_4 &= \sin(v_3) \\ v_5 &= \text{pow}(v_2, 2) \\ v_6 &= \cos(v_5) \\ v_7 &= v_4 + v_6 \\ y &= v_7 \end{aligned}$$

Die Abhängigkeiten dieser Berechnungen entsprechen dem Graphen in Abb. 5.2.

Durch diese Darstellungsweise wird sichtbar, dass man jeden Knoten des Graphen, also jedes (Zwischen-) Ergebnis, als Verkettung schreiben kann:

$$v_7 = \varphi_7(v_4, v_6) = \varphi_7(\varphi_4(v_3), \varphi_6(v_5)) = \varphi_7\left(\varphi_4(\varphi_3(v_0, v_1)), \varphi_6(\varphi_5(v_2))\right) \tag{5.13}$$

Jedes (Zwischen-)Ergebnis eines numerischen Programms ist also die Verkettung aller Operationen, die zum entsprechenden Knoten im Graph führen. Daher kann man die Ableitung der Ausgabewerte des Programms mittels Kettenregel berechnen.

5.5 Vorwärtsmodus

Durch die Anwendung der Kettenregel auf die Zwischenergebnisse an den Knoten des oben beschriebenen Graphen erhält man eine Vorschrift wie in jedem Rechenschritt auch

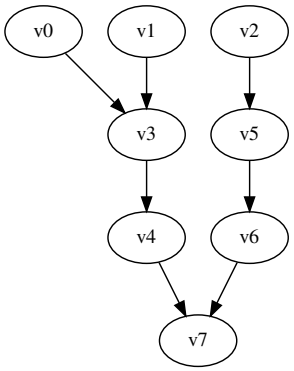


Abbildung 5.2: Berechnungsgraph der Prozedur f

Ableitungen berechnet werden können. Man kann nun jede Variable (=Knoten des Graphen) erweitern, dass sie nicht nur den aktuellen Wert v_i sondern noch eine zusätzliche Ableitungsinformation $\dot{v}_i$ speichert. Anschließend definiert man zu jeder Operation

$$v_j = \varphi_j(v_i)_{i \prec j} \quad j = n, ..., n+p+m-1 \tag{5.14}$$

noch die tangentiale Ableitung

$$\dot{v}_j = \sum_{i \prec j} \frac{\partial \varphi_j}{\partial v_i} \cdot \dot{v}_i \quad j = n, ..., n+p+m-1 \tag{5.15}$$

Zur Berechnung der Ableitung an jedem Knoten benötigt man also nur die Werte v_i den direkten Vorgänger, sowie die zugehörigen Richtungsableitungen $\dot{v}_i$. Die Werte $\dot{v}_j$ entsprechen den Ableitungen des j-ten Zwischenergebnisses nach der gewählten Ableitungsrichtung $\dot{x}$.

Hierdurch wird aus einer Prozedur, welche $y = f(x)$ berechnet, eine erweiterte Prozedur $\dot{f}$, welche neben dem eigentlichen Funktionswert auch eine Richtungsableitung berechnet

$$(y, \dot{y}) = \dot{f}(x, \dot{x}). \tag{5.16}$$

Mit

$$y = f(x) \tag{5.17}$$

$$\dot{y} = \frac{\partial f}{\partial x} \dot{x} \tag{5.18}$$

Die zweite Zeile kann dabei komponentenweise berechnet werden, ohne dass die vollständige Jacobi-Matrix der Prozedur aufgebaut oder gespeichert werden muss.

Die Definition der Richtung der Ableitung $\dot{x}$ geschieht durch die Wahl eines konstanten Vektors $\dot{x} \in \mathbb{R}^n$ und der Belegung der Ableitungen auf den Eingangsvariablen $\dot{v}_i =$

$x_i \quad i = 0..n-1$. Um nach einer bestimmten Eingangsvariable abzuleiten, wählt man $x = e_i$, den i-ten Einheitsvektor.

Dieser Modus kann in ein Programm integriert werden, indem jeder Variablen eine differenzierte Variable zugeordnet wird und zwischen den Rechenoperationen die differenzierten Operationen eingefügt werden. Durch Belegung der Tangenten-Werte an den Eingabevariablen wird die Richtung der Ableitung bestimmt. Während das Programm durchlaufen wird, wird dann zu jedem Zwischenergebnis v_i die gewünschte Richtungsableitung berechnet. Am Ende des Programmlaufs stehen daher für alle Ausgabegrößen Richtungsableitungen zur Verfügung. Durch einen Vorwärts-Lauf im tangentialen Vorwärts-Modus wird also bezüglich einer Richtung die Ableitung aller Ausgabegrößen berechnet.

5.6 Rückwärtsmodus

Der Rückwärtsmodus der automatischen Differentiation erlaubt es, Matrix-Vektor Produkte der Jacobi-Matrix eines Programms auf folgende Weise zu berechnen:

$$\bar{x} = \left(\frac{\partial f}{\partial x}\right)^T \bar{y} \tag{5.19}$$

Der Rückwärtsmodus basiert ebenfalls auf der Kettenregel, die Ableitungen werden jedoch rückwärts durch den Abhängigkeitsgraphen des Programms transportiert: Zunächst werden die Ergebnisse aller primalen Operationen berechnet:

$$v_j = \varphi_j(v_i)_{i \prec j}, \quad j = n, ..., n+p+m-1 \tag{5.20}$$

Danach können Werte die adjungierten Variablen Berechnet werden:

$$\bar{v}_i = \sum_{j:i \prec j} \frac{\partial \varphi_j}{\partial v_i} \bar{v}_j, \quad i = n+p-1, ..., 0 \tag{5.21}$$

Diese Summe kann nur aufgelöst werden, wenn man Rückwärts durch den Funktionsgraphen läuft. Dies wird verständlich, wenn man beachtet, dass $\bar{v}_i$ im Berechnungsgraphen zum Knoten i gehört. Um dort $\bar{v}_i$ zu berechnen benötigt man jedoch die adjungierten Werte aller Nachfolger im Graphen, also $\bar{v}_j$ mit $j : i \prec j$.

Gleichzeitig benötigt man zur Bildung der Jacobi-Matrix der Operation den primalen Wert am aktuellen Knoten, welcher nur im Vorwärtsdurchlauf berechnet werden kann, da er von seinen Vorgängerknoten abhängt.

Um die Operationen in umgekehrter Reihenfolge durchlaufen zu können, muss man daher im primalen Durchgang Ergebnisse für den Rückwärtslauf speichern.

Ein möglicher Ansatz ist es, im Vorwärtslauf die Jacobi-Matrix $\frac{\partial \varphi_j}{\partial v_i}$ der Elementaroperationen zu berechnen und zu speichern[1]. Da die Speicherung und der Abruf der

[1] Es existieren noch andere Speicherverfahren, welche effizienter sein können. Deren Darstellung ist jedoch für das Verständnis der Grundprinzipien hier nicht nötig

Jacobi-Matrizen nach dem LIFO-Prinzip (Last-In First-Out) erfolgen kann, eignet sich eine sequentielle Datenstruktur, welche in Analogie zu Bandlaufwerken im AD-Kontext Band genannt wird.

Der Ablauf des Rückwärts-Modus algorithmischer Differentiation besteht entsprechend der bereits erläuterten Technik aus zwei Phasen:

1. Speichern der Jacobi-Matrizen auf dem Band
2. Transport eines Vektors adjungierter Variablen durch das Band

Durch die Wahl der adjungierten Variablen auf den Ausgabegrößen wird die Ableitungsrichtung vorgegeben. Am Ende des Rückwärtslaufs liegen dann die Ableitungen der gewählten Ausgaberichtung bezüglich aller Eingabevariablen des Programms vor.

5.7 Implementierungstechniken

Wie kann man nun in der Praxis ein gegebenes Computerprogramm so erweitern, dass zwischen jeder ursprünglichen Elementaroperation (1.) die Berechnung ihrer Jacobi–Matrix stattfindet und (2.) der Transport der tangentialen Größen $\dot{v}$ vorwärts oder (3.) der Transport der adjungierten Größen $\bar{v}$ rückwärts durch das Programm stattfinden kann? Hierzu haben sich drei Verfahren etabliert:

1. Quelltext–Transformation
2. Operator–Überladen
3. Expression Templates

5.7.1 Quelltext–Transformation

Die naheliegendste Methode ist dabei sicher die Quelltext–Transformation, welche einen gegebenen Quelltext einliest, parst, den Syntaxbaum um die differenzierten Anweisungen erweitert und anschließend in Form eines erweiterten Quellcodes herausschreibt. Eine prominente Implementierung dieser Technik ist Tapenade [116]. Dieses erlaubt Programme der Sprachen Fortran und C zu differenzieren. Quelltext-Transformation kann neben der zu differenzierenden Elementaroperation auch deren Kontext erfassen und auf dessen Basis Optimierungstechniken für den adjungierten Code umsetzen. Der Vorteil der Quelltext-Transformation ist, dass der Quelltext adjungierter Routinen gelesen, verändert und diese als Bausteine verwendet werden können. Daher lässt sich diese Vorgehensweise sehr gut mit manueller Adjungierung kombinieren. Es ergeben sich jedoch auch Schwierigkeiten. Diese stammen vor allem daher, dass das AD-Werkzeug den gesamten Quelltext nur statisch, also ohne ihn auszuführen, bearbeitet.

Durch reines Betrachten des Quelltexts ist der Kontroll- und Datenfluss eines Programms jedoch nicht immer ersichtlich. Durch direkte Speicheroperationen wie Zeiger, Funktionszeiger, dynamische Bibliotheken u.ä. kann Beispielsweise in der Sprache C im allgemeinen erst zur Laufzeit festgestellt werden, welche Daten tatsächlich gelesen oder verändert werden und welche Unterfunktionen aufgerufen werden.

5.7.2 Operator–Überladen

Einige Programmiersprachen erlauben es, benutzerdefinierte Datentypen zu implementieren und für diese die Bedeutung aller Rechenoperatoren neu zu definieren. Operator-Überladen ist unter anderem in den Sprachen Fortran 90 und C++ möglich.

Die Umsetzung bedarf eines eigenen Datentyps, in C++ wird dies durch eine Klasse realisiert. Für diese definiert man alle mathematischen Elementaroperationen der Sprache neu. Da die Definition einer existierenden Funktion für neue Datentypen im C++–Kontext Überladen genannt wird, spricht man bei dieser Technik vom Operator-Überladen.

Hier wird also die AD-Logik durch die Änderung des Berechnungsdatentyps in das bestehende Programm „injiziert“. Das eigentliche differenzierte Programm wird dann durch den Compiler erzeugt, indem dieser die überladenen Funktionen dort aufruft, wo im ursprünglichen Code die primalen Operationen standen.

Diese Vorgehensweise hat den Vorteil, dass sie durch den gleichen Compiler durchgeführt wird, der den primalen Quelltext übersetzt. Es werden daher alle Sprachkonstrukte unterstützt, sofern der Compiler es für diese benutzerdefinierte Klassen und überladene Operatoren erlaubt.

5.7.3 Expression Templates

Expression Templates bauen auf dem Ansatz des Operator–Überladens auf und verbessern diesen. Da sie jedoch einen erheblichen Unterschied in der Implementierung des AD-Werkzeugs bedeuten, werden sie hier als eigene Implementierungstechnik beschrieben.

Bei den Erläuterungen zum Operator–Überladen ist zu sehen, dass tatsächlich für jede Elementaroperation eine überladene Operation aufgerufen wird. Hieraus entstehen zwei Probleme: Zunächst bedeutet jede Ausführung einer Elementaroperation nun einen Funktionsaufruf, welche einen kleinen Zusatzaufwand bedeutet. Zweitens wird in jeder dieser Funktionen ein Zwischenergebnis erzeugt, was die Anzahl der Speicherzugriffe signifikant erhöht. Ein optimierender Compiler wird die Funktionsaufrufe nicht tatsächlich durchführen, sondern den Funktionsinhalt an der Stelle ihres Aufrufs einfügen (inlining). Auch Zwischenvariablen können im Prinzip durch einen optimierenden Compiler entfernt werden.

Allerdings gelingt dies dem Compiler nicht in jedem Fall. Im Rückwärtsmodus kommt hinzu, dass für jede Elementaroperation ein zusätzlicher Zugriff auf das Band und somit auf den Speicher stattfindet. Es wäre oft effizienter, einen ganzen Ausdruck, also eine ganze Programmzeile, am Stück zu differenzieren, da so der Zusatzaufwand durch Funktionsaufrufe und Speicherzugriffe reduziert werden könnte. Dies kann mit Expression Templates erreicht werden.

Expression–Templates basieren auf der generischen Template–Programmierung [117] mit der in C++ parametrisierte Vorlagen (Templates) für Datentypen und Funktionen angelegt werden können. Der Compiler kann dann für verschiedene Werte der Template–

Parameter zur Übersetzungszeit konkrete Implementierungen der Funktion oder des Datentyps erzeugen und diese so optimieren, als wären sie explizit programmiert worden.

Der Kunstgriff der Expression–Templates [118] besteht darin, einen zusammengesetzten Berechnungsausdruck als templatisierte Klasse aufzufassen. Für jede Elementaroperation wird eine spezialisierte Klasse implementiert, die als Operanden wieder beliebige Expression-Klassen zulässt.

Auf diese Weise können Ausdrücke im Quelltext als Hierarchie von Klassen zur Übersetzungszeit abgebildet werden, welche dem Berechnungsgraphen des zusammengesetzten Ausdrucks entsprechen.

Der Aufruf der Methode zur Berechnung der Jacobi-Matrix des Ausdrucks kann dann rekursiv durch den Objektbaum absteigen und so die gesamte Jacobi-Matrix des Ausdrucks generieren. Aufgrund der statischen Typisierung der Sprache C++ ist die Struktur des Objektbaums bereits zum Übersetzungszeitpunkt bekannt. Somit kann ein Compiler den Code zur Generierung der Jacobi-Matrix eines Ausdrucks schon beim Übersetzungsvorgang erzeugen und diesen in nachfolgende Optimierungsvorgänge einbeziehen.

Implementierungen, die mit dieser Technik arbeiten, sind Sacado [119], Adept [120], DCO [121] und CodiPack [122]. Zur Rückwärtsdifferentiation von TRACE wurden im Rahmen dieser Arbeit DCO und CodiPack eingesetzt.

5.8 Eigenschaften algorithmischer Differentiation

Im Folgenden sollen einige Eigenschaften Algorithmischer Differentiation betrachtet werden, welche für die Implementierung der adjungierten Löser relevant sind.

5.8.1 Berechnung mit Maschinengenauigkeit

Bei der Vorstellung finiter Differenzen als Ableitungstechnik wurde auf die beiden Fehlerquellen dieser Technik eingegangen: Abbruch- und Auslöschungsfehler. In der algorithmischen Differentiation kommen hingegen analytisch hergeleitete Ableitungsoperationen zum Einsatz, so dass ein Resultat von Griewank (vgl. 3.4 in [17]) lautet: Die Jacobi-Vector Produkte, welche im Vorwärts oder Rückwärts-Modus berechnet werden, sind exakte Ableitungen für eine primale Berechnungsvorschrift in der jede Elementaroperation durch Rundungseffekte im Bereich der Maschinengenauigkeit (hier $\varepsilon \approx 1 \times 10^{-16}$) gestört wird. Der dort auftretende numerische Berechnungsfehler führt also nicht zu ungenaueren Ableitungen.

5.8.2 Rückwärtsmodus der Differentiation

Algorithmische Differentiation erlaubt die eben beschriebene Differentiation im Rückwärtsmodus. Bei der Differentiation im Rückwärtsmodus hängt der Aufwand zur Berechnung der Jacobi-Matrix einer Funktion wesentlich nur von der Zahl der Ausgabegrößen ab. Dies macht die Berechnung von Sensitivitäten für eine Vielzahl von Anwendungsfällen erst effizient. Diese Form der Differentiation ist mit finiten Differenzen nicht möglich, bei ihr ist der Aufwand zur Berechnung der Ableitungen immer proportional zur Zahl der

Eingabeparameter. Rückwärtsdifferentiation kann nur durch händisches Implementieren oder AD erreicht werden. Dabei ist anzumerken, dass einzelne Terme, die bei der händisch implementierten Rückwärtsdifferentiation auftreten, durch Vorwärtsdifferentiation effizient berechnet werden können.

5.8.3 Automatisierbare Erzeugung und Konsistenzerhaltung

Ist ein Programm einmal für die Differentiation mittels AD vorbereitet, lässt sich die eigentliche Differentiation automatisiert wiederholen. Dies ermöglicht die einfachere Erhaltung der Konsistenz. Wir bezeichnen mit P die primale Berechnungsvorschrift und $\bar{P}$ die zugehörige adjungierte. Bei einer Änderung der primalen Berechnung von P zu P_{mod} muss $\bar{P}_{\text{mod}}$ erzeugt werden.

Bei der händischen Differentiation müsste zunächst erkannt werden, dass $P_{\text{mod}} \neq P$ ist. Für reale Programme kann dies eine eigene Herausforderung darstellen, da P aus zahlreichen Unterprogrammen bestehen kann und für jede Änderung eines beteiligten Quelltexts beurteilt werden muss, ob sich eine Veränderung der Berechnungsvorschrift ergibt. Anschließend müsste ein Entwickler $\bar{P}_{\text{mod}}$ herleiten und implementieren. Dies ist zum einen mit der Möglichkeit eines Fehlers behaftet, so dass ungewollt $\bar{P}_{\text{mod}'}$ erzeugt wird und zum anderen ist es zeitaufwändig, so dass zumindest für eine gewisse Zeit die primale Berechnung P_{mod} mit $\bar{P}$ inkonsistent differenziert wird.

Sind die Vorbereitungen von P für AD einmal manuell durchgeführt, kann die Erzeugung differenzierter Routinen durch ein Computerprogramm automatisiert bei jeder Änderung des Quelltexts durchgeführt werden. Typischerweise geschieht dies direkt vor oder während der Übersetzung des Quelltexts in Maschinenanweisungen. Ändert sich dabei P zu P_{mod} wird direkt die konsistente differenzierte Variante erzeugt, vorausgesetzt die Wahl der unabhängigen Variablen x und Ausgabegrößen y ändert sich nicht und die technischen Voraussetzungen für das AD-Werkzeug werden nicht verletzt.

5.8.4 Differentiation bei stückweiser Differenzierbarkeit

Jedes reale Programm enthält Anweisungen, welche nicht-differenzierbare Stellen enthalten. Beispiele sind Entscheidungen, Betragsfunktion, Wurzel etc. . Sie alle haben gemeinsam, dass die zugrundeliegenden Funktionen auf mehreren offenen Intervallen differenzierbar sind.

Bei der algorithmischen Differentiation wird die Ableitung an einer gegebenen Stelle, also für einen ganz bestimmten Satz von Eingabevariablen der Prozedur berechnet. Es kann gezeigt werden (vgl. Kap.14 in [17]), dass die Elementaroperationen, aus denen sich numerische Programme zusammensetzen, bis auf isolierte Stellen stabil analytisch differenzierbar oder stabil undefiniert sind. Stabil bedeutet in diesem Zusammenhang, dass kleine Änderungen der Eingabewerte die Differenzierbarkeit nicht beeinflussen.

Werden isolierte nicht-differenzierbare Stellen der Elementaroperationen Teil der primalen Auswertung, kann dies durch das AD-Werkzeug zur Laufzeit erkannt werden und durch spezielle IEEE Gleitkomma-Werte (Inf, NaN) in den Ableitungsberechnungen signalisiert werden.

Einen Sonderfall bildet das Nachschlagen von Werten in Tabellen - hier sind die einzelnen Ergebnisse im Sinne algorithmischer Differentiation konstant, selbst wenn eventuell anschließend zwischen Tabellenwerten interpoliert wird. In diesen Fällen muss die Ableitungsinformation zu den Tabellenwerten getrennt implementiert werden oder, falls möglich, die Tabellen-Werte durch eine Berechnung ersetzt werden.

5.8.5 Berechnungsaufwand

Durch AD erzeugte Ableitungsberechnungen haben häufig den Ruf, teuer bezüglich Speicherplatz und Rechenaufwand zu sein. Tatsächlich kann bei manueller Implementierung von differenzierten Routinen effizienterer Code erzeugt werden. In der Literatur zu AD finden sich einige theoretische Untergrenzen für den Berechnungsaufwand algorithmisch differenzierter Routinen unter gewissen Annahmen für den relativen Aufwand zur Durchführung von Elementarberechnungen (vgl. Kap. 4 in [17]). Die folgenden Zahlen basieren auf drei Annahmen: Es wird davon ausgegangen, dass nur Gleitkommavariablen und Operationen vorkommen. Es wird kein Zusatzaufwand berücksichtigt für die technische Umsetzung der AD, wie zum Beispiel Funktionsaufrufe des Operator-Überladen, geändertes Zugriffsverhalten durch geändertes Speicherlayout etc. . Ferner werden bestimmte Annahmen über das Kostenverhältnis primitiver Operationen untereinander getroffen wie sie typisch für verbreitete Rechnerarchitekturen sind.

Im Vorwärtsmodus der AD liegt dann die Laufzeit im Verhältnis zur primalen Berechnungsvorschrift, abhängig vom Typ der durchgeführten Operationen, im Intervall $[2, 5/2]$. Der benötigte Speicher verdoppelt sich, da jeder Variable genau eine tangentiale Variable zugewiesen wird.

Im Rückwärtsmodus hängt die Obergrenze des Aufwands für eine Gradientenpropagation sehr stark von den Annahmen über die Menge der Speicherzugriffe ab. Konservative Abschätzungen führen zu einem Laufzeitverhältnis im Intervall $[3, 5]$ (vgl. [123]). Für die Aufzeichnungsphase muss für jede Variable ein eindeutiger Index der entsprechenden Variable mitgeführt werden, was typischerweise zu einer Verdoppelung des benötigten Speichers führt. Dieser Speicheraufwand wird jedoch dominiert von dem zusätzlich nötigen Speicher für das Band, dessen Größe proportional zur Anzahl der Operationen im primalen Code ist. Bei der Auswertung im Rückwärtsmodus besteht der Speicheraufwand aus dem des Bands sowie einer adjungierten Variablen pro primaler Variable. Die tatsächliche Laufzeit für die Auswertung im Rückwärtsmodus wird im wesentlichen von der Geschwindigkeit dominiert, mit der das Band ausgewertet werden kann. Diese hängt davon ab, ob das Band noch in den Hauptspeicher des Rechners passt, oder auf langsamere Speichermedien ausgelagert werden muss.

5.9 Techniken zur Ressourcenreduktion

Wendet man algorithmisches Differenzieren im Rückwärtsmodus auf größere Programme an, kann die Größe des entstehenden Bands sehr schnell die Größe des verfügbaren Speichers überschreiten. Außerdem bedeutet ein größeres Band aufgrund begrenzter

Speicherdurchsatzraten in den meisten Fällen auch eine längere Auswertungszeit. Ressourcenoptimierung eines algorithmisch differenzierten Programms zielt daher auf die Reduktion der Band-Einträge. Die nachfolgend beschriebenen Techniken sind in ihren Grundzügen bereits im Lehrbuch von Griewank [17] beschrieben. Die technische Umsetzung für TRACE ist in [27] sowie [124] beschrieben.

5.9.1 Function–Checkpointing

Eine Möglichkeit zur Speicherreduktion besteht darin, eine Prozedur P mit vielen Operationen nur temporär auf dem Band zu speichern, wenn sie benötigt wird. Dies kann man erreichen indem während des Vorwärtslaufs beim Erreichen von P die Bandaufzeichnung pausiert wird. Anschließend wird die Prozedur P ausgeführt und es wird ein Vermerk auf dem Band gespeichert, dass hier Einträge der Prozedur P fehlen. Außerdem müssen die Werte aller Eingabegrößen der Prozedur auf dem Band gespeichert werden. Dieser Band-Eintrag heißt Function–Checkpoint. Nach Beendigung der Prozedur wird die Band-Aufzeichnung fortgesetzt.

Erreicht die Rückwärtsauswertung die vermerkte Stelle, wird die Auswertungl pausiert, die Eingabegrößen von P gesetzt und P wird ausgeführt, wobei jetzt die Operationen aufgezeichnet werden. Daraufhin werden diese Operationen ausgewertet und die Band-Einträge von P verworfen.

In dieser Form stellt diese Technik noch keine Speicherreduktion dar, da man zu einem Zeitpunkt das gesamte Band, inklusive der Einträge von P im Speicher halten muss. Es wird sogar noch zusätzlicher Speicher benötigt, da man alle Eingangsgrößen, von denen P direkt abhängt, zusätzlich speichern muss, um diese während der Rückwärtsauswertung aufrufen zu können. Im Folgenden sei $S()$ der Speicherbedarf, und Ω das gesamte Programm, wobei P ein Teilgraph von Ω ist. Sei ferner C ein Function–Checkpoint. Dann ist der Speicherbedarf nach dem Vorwärtslauf

$$S(\Omega) - S(P) + S(C). \tag{5.22}$$

Gelangt man im Rückwärtslauf an die Stelle von P, wird erst dann das Band für die Prozedur aufgezeichnet. Der Speicherbedarf ist dann

$$S(\Omega) - S(P) + S(C) + S(P) > S(\Omega). \tag{5.23}$$

Zu einer Speicherreduktion führt diese Technik erst, wenn mehrere Function–Checkpoints existieren. Dies kann der Fall sein, wenn P mehrfach durchgeführt wird, oder aber wenn noch andere Prozeduren mit Function–Checkpoints behandelt werden. Bezeichnen wir alle Prozeduren, die mit Function–Checkpoints behandelt wurden, mit P_i respektive zugehöriger Checkpoints C_i dann ist der höchste auftretende Speicherverbrauch während der Rückwärtsausführung mit dieser Technik:

$$S(\Omega) - \sum_i S(P_i) + \sum_i S(C_i) + \max_i(S(P_i)). \tag{5.24}$$

Darüber hinaus muss berücksichtigt werden, dass nun die Zeit zum Aufzeichnen des Bands der Funktion bei jeder Auswertung des Bands jedesmal dann anfällt, wenn die

Prozedur auf dem Band vorkommt. Daher erhöht sich die Ausführungszeit um $(N-1) \cdot T(P_i)$, wenn $T()$ die Zeit bezeichnet, die benötigt wird, um das Band einer Funktion aufzuzeichnen und N die Anzahl ausführengen der Prozedur P darstellt.

Zur praktischen Umsetzung dieser Technik müssen die Eingabegrößen von P auf dem Band abgelegt werden können. Die Prozedur P muss daher um Code ergänzt werden welcher dies erledigt. Bei AD-Werkzeugen auf Basis von Operator-Überladen oder Expression-Templates muss dieser Schritt manuell implementiert werden.

5.9.2 Prä-Akkumulierung

Eine ähnliche Technik ist die Prä-Akkumulierung von Jacobi-Matrizen. Sie basiert darauf, dass das Band für eine Prozedur sehr lang sein kann, ihre Ableitung aber als relativ kleine Jacobi-Matrix dargestellt werden kann. Dies ist der Fall, wenn die Funktion nur wenige Ein- und Ausgabevariablen besitzt, aber verhältnismäßig viele Operationen durchführt.

In diesem Fall kann man während der Aufzeichnung im primalen Lauf des Programms alle Einträge, die zu der Prozedur gehören, zu einer Jacobi-Matrix zusammenfassen und diese auf dem Band speichern.

Hierfür müssen Ein- und Ausgabegrößen der Prozedur markiert werden und die Prozedur so ergänzt werden, dass sie bei der Aufzeichnung des Bands ihre Jacobi-Matrix berechnet. Die Berechnung der Teil-Jacobi-Matrix kann sowohl vorwärts- als auch adjungiert durchgeführt werden. Im primalen Lauf wird am Beginn von P die aktuelle Bandposition markiert. Die Prozedur wird ausgeführt und ihre Operationen auf das Band geschrieben. Anschließend wird nacheinander für jede Ausgabevariable das Band der Prozedur rückwärts oder für jede Eingabevariable vorwärts durchlaufen. Dabei wird die entsprechende tangentiale oder adjungierte Variable mit 1 und alle anderen Variablen mit 0 belegt. Man erhält die Teil-Jacobi-Matrix Zeilen- oder Spaltenweise.

Anschließend werden die Bandeinträge von P gelöscht und stattdessen die nun vorliegende Jacobi-Matrix hinterlegt. Es ergibt sich zunächst eine Speicherreduktion um die Länge des Bands der Prozedur abzüglich der Größe der Jacobi-Matrix

$$\Delta S = S(P) - S(N_{\text{in}} \cdot N_{\text{out}}) \tag{5.25}$$

Wobei N_{in} und N_{out} die Anzahl von Ein- respektive Ausgabegrößen von P bezeichnet. Die Zeit für die Aufzeichnung des Bands wird durch diese Technik vergrößert, die Zeit für die Auswertung des Bandes verkürzt sich jedoch, da nun weniger Einträge abgearbeitet werden müssen. Dies ist besonders für die Fixpunkt-Rückwärts-Iteration vorteilhaft, da hier nur ein Band aufgezeichnet wird, dieses aber sehr oft ausgelesen wird.

5.9.3 Ausnutzung von Wissen über die implementierte Funktionalität

Bestimmte Operationen, insbesondere im Bereich der linearen Algebra, erzeugen viele Band-Einträge und besitzen eine große Jacobi-Matrix. Jedoch lassen sich diese effizient rückwärts differenzieren. Eine Sammlung von rückwärts differenzierten Ausdrücken für solche Probleme beschreibt Giles [112]).

In der Regel sind AD-Werkzeuge jedoch nicht in der Lage zu erkennen, dass für eine Prozedur eine effizient implementierte Differentiationsvorschrift existiert[2].

In diesen Fällen kann es effizienter sein, adjungierte Routinen per Hand zu schreiben. Hierfür muss das AD-Werkzeug ermöglichen, bei der Rückwärts-Auswertung zu stoppen und eine solche Hand-adjungierte Routine aufzurufen, welche die adjungierten Werte für die Eingangsvariablen der Ursprungsfunktion belegt.

Mit diesem Ansatz kann die Größe des Bands um $S(P)$ reduziert werden. Ob sich hierdurch auch eine Laufzeit-Verkürzung des Rückwärts-Laufs ergibt, hängt jedoch von der Laufzeit der Hand-adjungierten Routine ab.

Diese Vorgehensweise birgt jedoch die gleichen Schwierigkeiten in sich, wie bereits beschrieben: Die von Hand adjungierten Routinen können aufgrund von Fehlern inkonsistent sein, oder durch Änderungen an den primalen Routinen inkonsistent werden.

[2] Prinzipiell ist dies mit dem Source-Code-Transformation Ansatz möglich und wird auch aktiv erforscht. Für Operator-Überladen und Expression-Templates existieren keine automatisierten Ansätze.

6 Gegenüberstellung zweier Implementierungsansätze

Im Folgenden sollen nun zwei konkrete Implementierungen diskret adjungierter Löser für den Strömungslöser TRACE genauer betrachtet werden. Diese folgen zwei verschiedenen Vorgehensmodellen, dem Top-Down-Prinzip respektive dem Bottom-Up-Prinzip. Zunächst werden die Vorgehensweisen vorgestellt und anschließend die beiden Implementierungen des adjungierter Lösers besprochen. Hiernach folgt eine Diskussion der Eigenschaften im Vergleich beider Löser.

6.1 Vorgehensweisen zur Erstellung adjungierter Löser

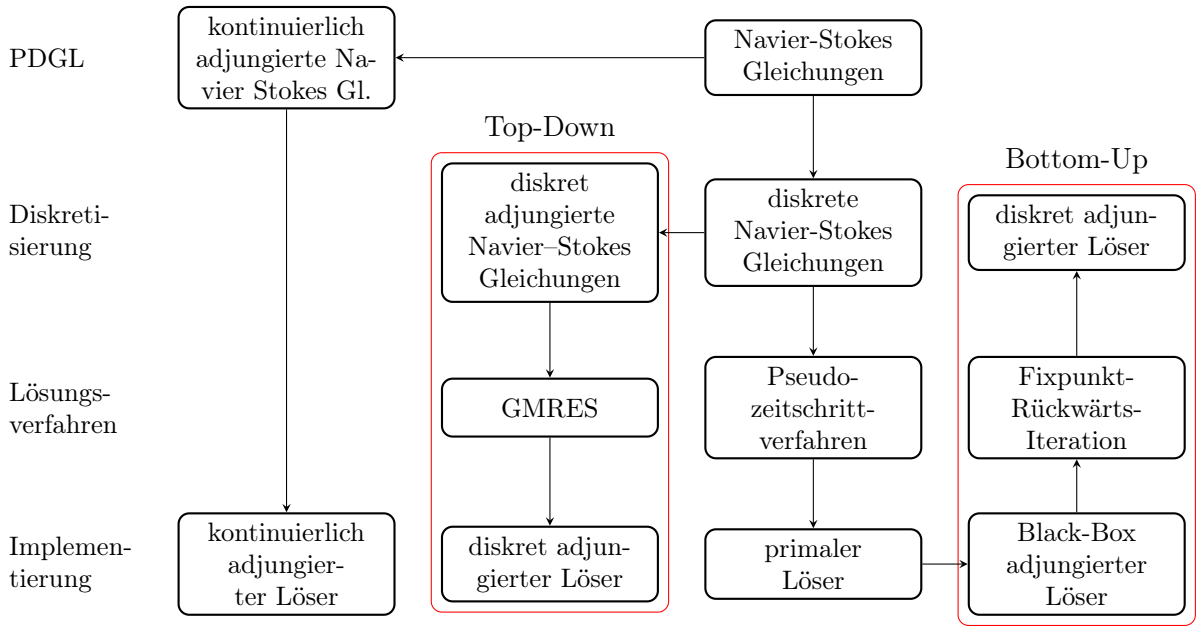


Abbildung 6.1: Schematische Darstellung der Vorgehensweisen zur Erstellung adjungierter Löser

Die Implementierung eines Strömungslösers kann man wie folgt untergliedern: Zunächst werden die beschreibenden Differentialgleichungen des zu simulierenden Problems formuliert. Darauf werden Diskretisierungsverfahren angewandt. Für die diskretisierten Gleichungen wird ein Lösungsverfahren auf Basis numerischer Algorithmen formuliert.

Diese werden schlussendlich mittels einer Programmiersprache in ein Simulationsprogramm umgesetzt.

Auf jeder dieser Stufen kann Adjungierung stattfinden. Eine Adjungierung der Differentialgleichungen mit anschließender Diskretisierung und numerischer Lösung wird kontinuierliches Adjungiertenverfahren genannt. Alle Verfahren, die nach der Diskretisierung ansetzen, heißen diskret adjungierte Verfahren.

Für die Adjungierung auf Ebene des Diskretisierungsverfahrens wird in der Literatur oft die manuelle Differentiation angewandt. Sie folgt dabei typischerweise der Herleitung wie sie in Abschnitt 4.2 dargestellt ist. Die Dünnbesetztheit der Residuen-Jacobi-Matrix wird durch manuelle Differentiation ausgenutzt, d.h. das Muster wie die Flüsse einer Zelle von Nachbarzellen abhängt wird händisch differenziert. Für die Differentiation der zugrundeliegenden Routinen wird dann häufig algorithmische Differentiation angewandt [23, 24, 25, 28, 125]. Nach Zusammensetzen der Residuen-Jacobi-Matrix, bzw. Erstellung von Routinen zur Berechnung von Matrix-Vektor Produkten mit dieser Matrix, wird diese als lineares Gleichunggsystem entsprechend Gl. 4.20 mit Standardverfahren gelöst.

Prinzipiell ist manuelle Differentiation auch auf der untersten Abstraktionsebene möglich. Dies würde bedeuten, den gesamten implementierten Löser anhand seines Quelltexts manuell zu differenzieren. Dies ist für Programme vom anvisierten Umfang und Komplexität zu aufwändig und zu fehleranfällig. Die Adjungierung gesamter Programme als Quelltext ist die Domäne der Werkzeuge zur algorithmischen Differentiation.

Das wesentliche Argument für die Adjungierung auf Ebene der Differentialgleichungen sind der größere Verständnisgewinn über die mathematischen Eigenschaften des adjungierten Lösers. Auch können durch dieses Verfahren sehr effiziente adjungierte Löser implementiert werden, besonders wenn bereits optimierte Bausteine des primalen Lösers bei der Implementierung wiederverwendet werden können.

Das Hauptargument für das Ansetzen auf den tiefer liegenden Schichten ist jedoch die Konsistenz. Ein adjungierter Löser zur Gradientenberechnung in der Optimierung wird in den allermeisten Fällen passend zu einem bereits existierenden Strömungslöser entwickelt. Die berechneten Ableitungen müssen zu den Ergebnissen dieses primalen Lösers passen. Durch die Adjungierung auf höheren Ebenen lässt man Details tiefer liegender Abstraktionsebenen aus, welche jedoch für die Ergebnisse des primalen Lösers und somit auch für die Konsistenz relevant sind.

Adjungiert man beispielsweise die Navier-Stokes Gleichungen im kontinuierlichen Verfahren, muss man zur Diskretisierung der adjungierten Gleichung das adjungierte Pendant der primalen Flussdiskretisierung finden. Da die Wahl der Flussdiskretisierung im primalen Löser einen erheblichen Einfluss auf das Ergebnis haben kann, besteht hier das Potential zu Inkonsistenz zwischen primalem und adjungiertem Löser. Selbst wenn die grobe Form der Flussdiskretisierung richtig adjungiert wurde, so müssen wichtige Details wie Flussberechnung an Rändern und Limiter-Funktionen auch konsistent adjungiert werden. Darüber hinaus können sich in der konkreten Implementierung immer Details verbergen, welche sich nicht leicht auf den jeweils höheren Abstraktionsebenen abbilden lassen und dennoch einen Einfluss auf die Ergebnisse haben und somit auch für die Gradientenberechnung relevant sind.

Beispielsweise zeigen Nadarajah und Jameson, dass die diskret adjungierten Gradienten bessere Übereinstimmung mit finiten Differenzen komplexer Schrittweite zeigen als kontinuierlich adjungierte [126]. Giles stellt fest, dass die numerische Glättungseigenschaft des Diskretisierungsschemas exakt adjungiert werden muss, um korrekte Ergebnisse in Gegenwart von Strömungsdiskontinuitäten (Stößen) zu erhalten [127].

Die Wahl der Vorgehensweise muss also im Spannungsfeld zwischen Abstraktion, Verständnisgewinn und Effizienz auf der einen Seite und Konsistenz auf der anderen Seite getroffen werden. Für den Einsatz der Optimierung ist die exakte Konsistenz der Gradienten zu den primalen Ergebnissen besonders wichtig.

Alle bis hier diskutierten Ansätze bewegen sich von der höheren Abstraktionsebene hin zur konkreten Implementierung und können somit Top-Down Ansätze genannt werden.

Ein alternatives Vorgehen besteht darin, algorithmische Differenzieren im Rückwärtsmodus auf den gesamten Löser anzuwenden, um einen vollständig konsistenten adjungierten Löser zu erhalten. Dieser wird jedoch zu ineffizient sein, um praktisch eingesetzt zu werden (vgl. Müller und Cusdin [13]). Dies liegt hauptsächlich daran, dass viele numerischen Verfahren iterativ durchgeführt werden und bei algorithmischer Differenzierung auch diese Iterationen differenziert werden, obwohl dies unnötig wäre. Effizientere Implementierungsweisen sind auf Ebene des Quelltexts schwer zu sehen, können jedoch leicht mit dem Verständnis höherer Abstraktionsschichten umgesetzt werden. Da dieser Ansatz auf der niedrigsten Abstraktionsebene startet und sukzessive Verbesserungen aus höheren Ebenen einbezieht, wird er im Folgenden Bottom-Up Ansatz genannt.

6.2 Top-Down Vorgehen

Der hier betrachtete Löser adjointTRACE wurde beginnend auf der Ebene der diskretisierten Navier-Stokes Gleichungen nach dem Top-Down Prinzip entworfen. Die Implementierung beschreiben Frey et al. [14]. Die Autoren kombinieren manuelle Adjungierung und finite Differenzen, um das adjungierte Gleichungssystem (s. Gl. 4.20) aufzustellen, welches dann als lineares Gleichungssystem iterativ gelöst wird. Im Rahmen dieser Arbeit wurden die finiten Differenzen durch algorithmisches Differenzieren im Vorwärtsmodus ersetzt [28].

6.2.1 Aufstellen des adjungierten Gleichungssystems

Der besseren Lesbarkeit halber wird hier das adjungierte Gleichungssystem aus Gl. 4.20 wiederholt:

$$\left(\frac{\partial R}{\partial q}\right)^T \Psi = \left(\frac{\partial J}{\partial q}\right)^T \tag{6.1}$$

Die rechte Seite ist die Ableitung des Auswerteprozesses nach dem berechneten Strömungszustand. Aus der Vielzahl möglicher Bewertungsgrößen sind Auswertegrößen auf der Austrittsebene und Bilanzierungen zwischen Ein- und Austrittsebene der Stufe besonders relevant. Aus diesen wurde ein Satz besonders oft benötigter Auswertegrößen bestimmt, darunter Massenstrom, Totaldruckverhältnis, Isentroper Wirkungsgrad und

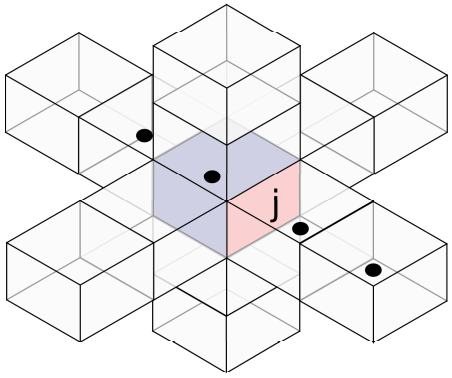


Abbildung 6.2: Abhängigkeitsmuster der konvektiven Flussberechnung in einem MUSCL-Schema zweiter Ordnung in einem strukturierten Rechennetz.

Drall. Diese wurden durch analytische Differentiation adjungiert und manuell implementiert. Detaillierte Beschreibungen für analytisch adjungierte Varianten relevanter Bewertungsgrößen für Turbomaschinen finden sich bei Frey et al. [128].

Die linke Seite des Gleichungssystems ist die Ableitung des Residuums nach dem Strömungszustand. Das Residuen der diskreten Navier-Stokes Gleichungen bilden eine Vektorwertige Funktion für die fünf Erhaltungsgleichungen. Sie lautet für die i-te Zelle

$$R_i = \frac{1}{V_i} \sum_{j \in \partial i} F_j(q) - S_i. \tag{6.2}$$

wobei q_i der mittlere Zustand der i-ten Zelle ist, $F_j(q) = F_j^c(q) - F_j^v(q)$ und j über die Seitenflächen ∂i der aktuellen Zelle läuft. Hierin wird die Bilanz der Flüsse F_j zusammen mit den Quelltermen S_i gebildet. Daraus erhalten wir eine Gleichung für die Residuen-Jacobi-Matrix:

$$\frac{\partial R_i}{\partial q_k} = \frac{1}{V_i} \sum_{j \in \partial i} \frac{\partial F_j}{\partial q_k} + \frac{\partial S_i}{\partial q_k} \tag{6.3}$$

6.2.2 Differenzierung der Flussberechnung

Für die Adjungierung der Flüsse muss man zwischen inneren und äußeren Flüssen unterscheiden: Randbedingungen und Gebietszerlegung werden im vorliegenden Fall durch zusätzliche Zellschichten um das eigentliche Rechengebiet herum abgebildet, welche Geisterzellen genannt werden. Während innere Flüsse zwischen Zellen auftreten, die innerhalb des Rechengebietes liegen, finden äußere Flüsse zwischen einer Zelle im Rechengebiet und einer Geisterzelle statt.

Innere Flüsse besitzen ein Abhängigkeitsmuster von Zuständen umgebender Zellen, welches durch das Diskretisierungsschema vorgegeben wird. Abbildung 6.2 zeigt beispielhaft die Abhängigkeit der Flussberechnung bei Diskretisierung konvektiver Flüsse mittels MUSCL-Schema zweiter Ordnung. Der Fluss durch die rot markierte Fläche hängt von Zuständen an den durch Punkten markierten Zellen ab. Dieses Abhängigkeitsmuster lässt

sich nutzen, um die Beiträge dieser Flüsse zur Residuen-Jacobi-Matrix dünnbesetzt zu speichern. Sie hat überall dort Einträge, wo der j-te Fluss vom k-ten Zustand abhängt. Da durch die fünf Grundgleichungen immer fünf Flüsse auf jeder Zellfläche und fünf Zustände in jeder Zelle auftreten, erhalten wir eine dünnbesetzte Block-Matrix, deren Einträge dichtbesetzte 5×5 Blöcke sind.

Für die Differenzierung innerer Flüsse werden zentrale finite Differenzen verwendet. Die Schrittweite wird dabei proportional zur Größe des jeweils zu differenzierenden Flusses gewählt. Die Entscheidung für die Differentiation mittels finiter Differenzen wurde aufgrund von Wartbarkeitsaspekten und der technischen Umsetzbarkeit algorithmischer Differentiation getroffen. Dies geschah vor allem im Hinblick auf verfügbare Werkzeuge und deren Anwendbarkeit auf die existierende Quelltextbasis des Lösers. Die Wahl der Differentiationstechnik wird in Abschnitt 6.2.5 weiter diskutiert.

Äußere Flüsse treten an den Rändern des Simulationsgebiets und an Unterteilungen innerhalb des Gebiets auf. Zu den Rändern des Simulationsgebiets zählen Ein- und Austritts und Wand-Randbedingung. Unterteilungen des Rechengebiets entstehen durch die Zerlegung des Gebietes in Blöcke zur Parallelisierung und durch die Kopplung rotierender und stehender Gebiete durch Mischungsebenen.

Zur Berechnung der äußeren Flüsse werden die Zustände in den Geisterzellen abhängig vom Zustand der inneren Zellen berechnet:

$$q_{\text{ext}} = \mathcal{T} q_{\text{int}} \tag{6.4}$$

Der Operator $\mathcal{T}$ ist im Allgemeinen nichtlinear und im Falle nicht-reflektierender Randbedingungen, Mischungsebenen, Austrittsbedingungen mit radialem Verteilungsgesetz nicht-lokal. Zudem repräsentiert er im Falle einer Block-Verbindung den Austausch von Zuständen von parallel laufenden Prozessen.

Die Flussberechnung ist eine Funktion innerer und äußerer Zustände

$$F(q_{\text{int}}, q_{\text{ext}}, x) \tag{6.5}$$

wobei für die Berechnung äußerer Flüsse spezielle Routinen für den jeweiligen Rand-Typ aufgerufen werden.

Für die Adjungierung von $\mathcal{T}$ und der Rand-Fluss-Routinen durch Frey et al. [129] wurde analytische Differentiation und manuelle Implementierung angewandt, da die beteiligten Berechnungen aufgrund von Relaxation der Randbedingungen ein konstanter Operator für die Randbedingungen im Quelltext nicht vorliegt. Da die Operationen am Rand häufig nicht-lokal und dichtbesetzt sind, wäre eine Differentierung mit finiten Differenzen und die Speicherung der resultierenden Jacobi-Matrix wenig effizient. Es wurden daher linearisierungen Operatoren der verschiedenen $\mathcal{T}$ für einen Fixpunkt hergeleitet und händisch implementiert. Diese Implementierungen werden während der Lösung des adjungierten Gleichungssystems angewandt.

Die Ableitungen der Quellterme $\partial S_i / \partial q_k$ wurden ebenfalls analytisch hergeleitet und manuell implementiert.

Auf die Differenzierung des Turbulenzmodells wurde in diesem Ansatz verzichtet, stattdessen wird die Annahme eines konstanten turbulenten Wirbelzähigkeitsfeldes (engl.

Constant Eddy Viscosity, CEV) verwendet. Der Einfluss auf die Konsistenz wird in Abschnitt 6.4.1 diskutiert.

6.2.3 Lösungsverfahren und Auswertung

Das Gleichungssystem wird mittels des Krylov-Verfahrens GMRES (generalized minimal residual method) [130] gelöst. Um dem Speicherverbrauch durch die wachsende Zahl und Größe der Unterräume zu begegnen, wird eine Schranke vorgegeben, nach welcher die Suche von der bisher besten Lösung neu gestartet wird (restarted GMRES).

Das Gleichunggsystem wird präkonditioniert, entweder durch „successive overrelaxation“ (SSOR) oder unvollständige LU-Zerlegung (ILU) mit einer begrenzten Zahl zusätzlicher Einträge außerhalb der ursprünglichen Besetzungsstruktur (ILU(p)).

Nach dem Lösen des Gleichungssystems folgt die Auswertung der Sensitivitäten nach Designparametern. Die im Strömungslöser verfügbaren Geometrieparameter entsprechen den Netzknoten des Volumennetzes x. Man will häufig jedoch übergeordnete Parameter α optimieren, die das Volumennetz jedoch eindeutig bestimmen $x = x(\alpha)$, wobei die Ableitung nur mittels finiter Differenzen in Form deformierten Netze $\delta x_j = x(\delta\alpha_j)$ repräsentiert wird:

$$\left(\frac{\partial J_i}{\partial \alpha_j}\right) = \Psi^T \left(\frac{\partial R}{\partial x}\delta x_j\right) \quad i = 1..m, j = 1..n. \tag{4.21}$$

Der Term $(\partial R/\partial x)\delta x_j$ wird dabei als ein Schritt aufgefasst, in welchem die Jacobi-Matrix nicht zunächst explizit aufgestellt werden muss, sondern nur Skalarprodukte mit ihr ausgewertet werden müssen, was mit erheblich weniger Speicheraufwand möglich ist. Die Auswertung wird mittels der zentralen Differenz

$$\frac{\partial R}{\partial x}\delta x \approx \frac{R(q, x + h\delta x) - R(q, x - h\delta x)}{2h} \tag{6.6}$$

ausgeführt.

6.2.4 Netzsensitivitäten

Die zuletzt dargestellte Berechnung hat den implementierungstechnischen Nachteil, dass man das primale R und den primalen Strömungszustand zu diesem Zeitpunkt erneut benötigt.

Die adjungierte Lösung Ψ entspricht der Ableitung des Residuums nach zusätzlichen Quelltermen der Erhaltungsgleichung. Ein direkter Erkenntnissgewinn aus adjungierten Lösungen ist möglich, aber schwierig [131, 132].

Für eine anschauliche Darstellung ist die Auswertung der Ableitung nach Oberflächenpunkten s günstig. Die Abhängigkeitskette lautet dabei $s \mapsto x \mapsto R$, mit Volumengitterkoordinaten x. Als Zwischenschritt wird die Ableitung des Residuums nach allen x gebildet:

$$\left(\frac{\partial J_i}{\partial \alpha_j}\right) = \left(\Psi^T \frac{\partial R}{\partial x}\right) \delta x_j \quad i = 1..m, j = 1..n \tag{6.7}$$

Dabei sind die $\Psi^T(\partial R/\partial x)$ Netz-Sensitivitäten, also die Abhängigkeit eines gewählten Funktionals von allen Gitterkoordinaten des Rechennetzes. Diese werden direkt im adjungierten Löser nach der Bestimmung von Ψ berechnet und hiermit multipliziert. Für die praktische Umsetzung [133] ist dabei wichtig, dass das Residuum des primalen Strömungslösers nur indirekt von den Gitterkoordinaten des Netzes abhängt:

$$R(q, \xi(x)), \tag{6.8}$$

wobei die Funktion $\xi(x)$ abgeleitete Geometriegrößen, wie Zellvolumina, Zellflächeninhalte, Normalenvektoren sind, welche nicht als eine abgeschlossene Routine berechnet sind, sondern an mehreren Stellen innerhalb der Initialisierung berechnet werden. Daher wird

$$\frac{\partial R}{\partial x} = \frac{\partial R}{\partial \xi}\frac{\partial \xi}{\partial x} \tag{6.9}$$

zweiteilig bestimmt: Die Ableitung von $\partial\xi/\partial x$ ist manuell bestimmt und implementiert, während $\partial R/\partial\xi$ mittels zentraler Differenzen approximiert wird.

Aus einer Reihe von Gründen kann es sinvoll sein, die Abhängigkeit der Netzkoordinaten von den Oberflächenpunkten s der Schaufeln und des Ringraums ebenfalls einzubeziehen. Differenziert man die Abhängigkeit $x(s)$ mit, so erhält man Ableitungen des Funktionals nach Oberflächenpunkten. Dies ist sinnvoll für die Interpretation der Ergebnisse, sowie die Abschätzung von Fertigungsabweichungen, sowie zur Reduktion des benötigten Speichers bei der Sensitivitätsauswertung bezüglich der Entwurfsparameter α, da $\frac{\partial R}{\partial s}$ eine Größenordnung kleiner ist als $\frac{\partial R}{\partial x}$. Dies kann beuspielsweise durch die Adjungierung eines Netzdeformationswerkzeugs erreicht werden [133]. Im Weiteren wird dies nicht weiter betrachtet.

6.2.5 Selektive Nutzung algorithmischer Differentiation

Der Approximierungsfehler der zentralen Differenzen für die Fluss-Funktionen wurde eingehend untersucht [71] und die Ergebnisse des adjungierten Lösers wurden für zahlreiche Anwendungsbeispiele verifiziert [14, 128, 53, 62, 94]. Ein zeitlinearisiertes Verfahren welches auf dem gleichen primalen Löser basiert, verwendet die gleiche Approximation der Residuen-Jacobi Matrix und wurde ebenfalls validiert [134]. Dennoch existiert ein Einfluss der Schrittweitenwahl auf das Ergebnis des adjungierten Lösers. Man kann nicht allgemein sicherstellen, dass die gewählte Heuristik zur Schrittweitenwahl für alle möglichen Konstellationen geeignet ist. Um die Schrittweitenproblematik zu eliminieren, wurden die finiten Differenzen zur Berechnung der Fluss-Jacobi-Matrix $\partial F/\partial q_{\text{int}}$ durch algorithmische Differentiation ersetzt [28]. Hierfür wird der Ansatz des Operator-Überladens im Vorwärtsmodus mit dem tangentialen Datentyp von ADOL-C verwendet.

Da die Operationen mit AD-Datentypen langsamer sind und mehr Speicher benötigen als solche in Standardarithmetik, sollen nur zu differenzierende Programmteile mit AD-Arithmetik durchgeführt werden. Insbesondere das für Laufzeit und Speicherbedarf maßgebliche GMRES-Verfahren muss nicht differenziert werden. Naheliegend wäre es daher, nur die Systemmatrix Ψ sowie eine rechte Seite $\partial J/\partial q$ im adjungierten Löser

zu berechnen und diese einem getrennt implementierten GMRES-Löser zu übergeben. Der Ressourcenbedarf zum Aufstellen des Gleichungssystems ist vernachlässigbar gegenüber den Kosten zum Lösen des Gleichungssystems. Jedoch soll, entsprechend des bereits beschriebenen Ansatzes, nur $\partial F/\partial q_{\text{int}}$ mit AD berechnet und gespeichert werden.

Da die Ableitung $\partial F/\partial q_{\text{ext}}$ Randbedingungen sowie Kommunikation beeinhaltet und außerdem dichtbesetzt ist, wird diese nicht gespeichert, sondern mittels manuell differenzierter Funktionen während jeder Iteration GMRES-Verfahrens bei Bedarf ausgewertet. Das bedeutet, das innerhalb des GMRES-Verfahrens spezielle, manuell differenzierte Routinen aufgerufen werden müssen, welche sich wiederrum aus Routinen und Datenstrukturen des primalen Lösers zusammensetzen. Eine Aufteilung des Lösers in zu differenzierende und nicht zu differenzierende Routinen ist daher technisch schwierig.

Ein Lösungsansatz hierzu ist die gleichzeitige Bereitstellung aller Routinen des primalen Lösers in AD-Arithmetik und Standardarithmetik.

Dies wird durch die ein- und ausschaltbare Überladung des gesamten Lösers mit dem Vorwärts-AD Datentyp erreicht. Somit können zwei Programme aus dem gleichen Quelltext erzeugt werden: Im ersten kann jede beliebige Routine vorwärtsdifferenziert werden, im zweiten wird mit Standardarithmetik ohne Zusatzaufwand gerechnet.

Es stellt sich jedoch das Problem, dass differenzierte und nicht-differenzierte Routinen in unterschiedlichen Programmen laufen und miteinander kommunizieren müssen. Außerdem soll der differenzierte Löser nur bei Bedarf gestartet werden und nur die tatsächlich benötigten Operationen durchführen, da er zusätzlich doppelt so viel Speicher benötigt wie der undifferenzierte Löser. Der Bedarf, zwei verschiedene Löser miteinander kommunizieren zu lassen, tritt auch bei der gekoppelten Simulation verschiedener physikalischer Simulationen, wie Strömungssimulation und Strukturmechanik auf. Hiervon inspiriert werden undifferenzierter und differenzierter Löser über das Message-Passing-Interface (MPI) miteinander gekoppelt, wobei zunächst nur der undifferenzierte Löser gestartet wird.

Bei Bedarf startet dieser den differenzierten Löser, um $\partial F/\partial q_{\text{int}}$ mittels AD-Datentyp zu berechnen, als dünnbesetzte Matrix im Standard-Datentyp bereitzustellen, und per MPI zurück zu kommunizieren. Der differenzierte Löser wird daraufhin beendet. Im undifferenzierten Löser werden die manuell differenzierten Routinen zur Berechnung der Quellterme $\partial S/\partial q$ und der rechten Seite $\partial J/\partial q$ ausgewertet, um das adjungierte Gleichungssystem fast vollständig aufzusetzen. Die noch fehlenden Anteile $\partial F/\partial q_{\text{ext}}$ werden während des Lösungsprozesses ausgewertet.

Diese Art der Kopplung hat den Vorteil, dass sie über bekannte Kopplungsverfahren die Differenzierung beliebiger Routinen des primalen Lösers ermöglicht und dennoch die Berechnung in unbeeinflusster Standardarithmetik erlaubt.

6.3 Bottom-Up Vorgehen

In dieser Vorgehensweise wird zunächst durch algorithmisches Differenzieren im Rückwärtsmodus des gesamten Quelltexts des Strömungslösers ein konsistenter adjungierter Löser erzeugt. Ein solcher Black-Box-Löser (im Sinne von Abs. 4.3) muss jedoch sämt-

liche Operationen aller Iterationen des primalen Lösers aufzeichnen und besitzt daher einen zu hohen Speicherverbrauch für praktische Zwecke. Aufgrund der algorithmischen Erstellung und Unabhängigkeit von Konvergenzannahmen ist er jedoch eine verlässliche Referenz. Ausgehend von dieser Referenz werden manuell Effizienz-Optimierungen implementiert, welche das Ergebnis nicht wesentlich ändern und sich daher stets verifizieren lassen.

Die im Folgenden beschriebene Implementierung ist in Kooperation mehrerer Personen unter anderem des Autors dieser Arbeit entstanden [27, 124, 135].

6.3.1 Austausch der Berechnungsdatentypen

Für die Bottom-Up Vorgehensweise wird zunächst mittels Operator-Überladen der gesamte Quelltext des Lösers rückwärts differenziert. TRACE ist in der Programmiersprache C geschrieben. Operator-Überladen ist in C zwar gar nicht möglich, jedoch in der Sprache C++ welche gelegentlich als Obermenge von C bezeichnet wird. Bei dieser Sichtweise sollte jedes C Programm auch ein gültiges C++ Programm sein. Dies ist jedoch leider nicht der Fall [136]. Es existieren verschiedene C Standards, welche Erweiterungen gegenüber dem ursprünglichen ANSI-C89 Standards sind. Der Quelltext des TRACE Strömungslösers ist Konform zum ISO-C99 Standard, welcher nur teilweise im, zur Zeit der Entwicklung verfügbaren, Standard ISO-C++11 abgebildet wird.

Da der primale Löser weiterhin in C entwickelt werden soll, wurde der gesamte Quelltext überarbeitet, so dass er der Schnittmenge beider Sprachstandards genügt [27, 28]. Dies ist unter Beachtung einiger Einschränkungen möglich, die Konformität kann automatisiert durch das Übersetzen mit Compilern für beide Sprachstandards überwacht werden.

Zur Umsetzung von AD wurde genutzt, dass alle numerischen Berechnungen bereits auf einer eigenen Typdefinition basieren, welche im Normallfall ein Alias für den Standard-Gleitkommatyp der Sprache C darstellt. Diese Typdefinition kann umgeschaltet werden, so das verschiedene AD-Datentypen eingebunden werden können. Dies wurde umgesetzt für die Vorwärts- und Rückwärts-Typen von ADOL-C [137], DCO [121] und CodiPack [122]. ADOL-C dient als technisch reifste Implementierung basierend auf der Technik des Operator-Überladens in C++. DCO und CodiPack sind hoch-optimierte Implementierungen auf Basis der C++ Expression-Templates.

Resultate und Leistungsmessungen in dieser Arbeit basieren auf CodiPack. Die Verwendung einer Klasse anstelle eines des primitiven Gleitkommadatentyps erzeugt weitere Einschränkungen an die erlaubten Programmierkonstrukte, deren Erläuterung an dieser Stelle zu technisch würde. Nachdem die nötigen Anpassungen einmal umgesetzt wurden, ist es daher sinnvoll mittels automatisierter Tests (Continuous Integration s. [138]) bei jeder Änderung des Quelltexts zu überwachen, dass der Quelltext sich noch mit allen gewünschten Datentypen übersetzen lässt.

Durch Umschalten des Datentyps und anschließendem Übersetzen des Quelltexts können daher verschiedene differenzierte Programme aus dem gleichen primalen Löser erzeugt werden. So lässt sich sicherstellen, dass die Effizienz des primalen Lösers nicht

durch AD-bedingten Zusatzaufwand beeinträchtigt wird, gleichzeitig jedoch jederzeit eine konsistente rückwärts-differenzierte Variante des Lösers erzeugt werden kann.

6.3.2 Aktive und passive Variablen

Für die algorithmische Differenzierung unterscheidet man aktive und passive Variablen. Variablen die im Berechnungsgraph von einer Eingangsvariable erreicht werden können und von denen eine Ausgangsvariable erreichbar ist, heißen aktive Variable. Inaktive Variablen sind nicht Teil dieser Abhängigkeit, sie können daher bei der Differenzierung vernachlässigt werden. Aktive Variablen müssen vom AD-Datentyp sein, damit die Ableitungen korrekt berechnet werden. Inaktive Variablen sollten vom Standard-Datentyp sein, um unnötige Zusatzberechnungen zu vermeiden. Da die Aktivität einer Variable jedoch abhängig von der konkreten Berechnung und den als Ein- und Ausgangsvariablen markierten Größen ist, nimmt man eher in Kauf zu viele Variablen als aktiv zu betrachten. Im Folgenden werden Variablen des AD-Datentyps als aktiv und Variablen des Standardatentyps als passiv bezeichnet.

Die AD-Datentypen sind so gestaltet, dass sie bei Verwendung einer passiven Variable in einem Ausdruck, der einer aktiven Variable zugewiesen wird, diese als aktiv betrachten. Dabei wird angenommen, dass die passive Variable konstant ist, also ihr zugehöriger Ableitungswert null ist. Die Zuweisung einer aktiven an eine passive Variable benötigt jedoch einen expliziten Konvertierungsaufruf. Dies ist sinnvoll, um das versehentliche unterbrechen der Abhängigkeitskette für die Differenzierung zu vermeiden.

Dies passiert häufig aus zwei Gründen: Dort wo Werte in Dateien oder auf dem Bildschirm ausgegeben werden und dort wo Berechnungen durch externe Bibliotheksfunktionen stattfinden. Im Falle von Ausgaben muss lediglich eine Konvertierung von aktiver nach passiver Variable durchgeführt werden. Da der Berechnungsbaum an einer Ausgabestelle ohnehin endet, kann dabei auch die Ableitungsinformation vernachlässigt werden. Für Ausgaberoutinen der C-Standardbibliothek (z.B. printf) kann dies generisch erfolgen. Händische Implementierung von aktiv nach passiv Konvertierungen ist dagegen bei nicht-Standard Ausgaberoutinen nötig - im betrachteten Löser beispielsweise bei der Schnittstelle zum Datenformat CGNS[1].

Komplizierter ist der Fall beim Aufruf externer Berechnungsroutinen. Ein Abschneiden des Ableitungsbaums durch Konvertierung in passive Datentypen hätte fehlerhafte Resultate zur Folge. Konvertieren in passive Variablen entspricht der Annahme, dass die Ableitungen der Resultate bezüglich der mathematischen Operationen in der Bibliothek null ist.

Es gibt verschiedene Möglichkeiten zur Behandlung externer Berechnungen: In einigen Fällen kann der Einfluss der Berechnungen in der Bibliotheksfunktion bewusst vernachlässigt werden. In diesem Fall kann man durch Konvertierung die Ableitung abschneiden, muss jedoch den resultierenden Fehler akzeptieren.

Kann der Einfluss einer Berechnung nicht als konstant angenommen werden, so kann man die entsprechende Bibliotheksfunktion durch eine eigene Implementierung ersetzen,

[1] CFD General Notation System (http://www.cgns.org/)

welche dann durch das AD-Werkzeug differenziert wird. Dieses Vorgehen ist nur für einfachere Funktionalitäten sinnvoll, da der Aufwand für Entwicklung und Fehlersuche, welchen die Bibliotheksentwickler bereits betrieben haben, erneut geleistet werden muss.

Alternativ kann man hand-adjungierte Routinen der Bibliotheksfunktion implementieren, sofern man die Berechnungen darin genau kennt. Dies setzt voraus, dass das AD-Werkzeug Unterstützung für hand-implementierte adjungierte Routinen bietet. Da diese Technik jedoch auch zur Effizienzsteigerung bei linearen Operationen verwendet werden kann, ist dies bei den hier bereits eingebundenen AD-Werkzeugen möglich.

Nicht zuletzt ist es möglich, sofern der Quelltext der externen Bibliothek ebenfalls als C++-Quelltext vorliegt, die vollständige Ersetzung des Datentyps auch innerhalb der Bibliothek vorzunehmen. So erhält man eine automatisch differenzierte Variante der Bibliothek, welche nahtlos in ein algorithmisch differenziertes Programm eingebunden werden kann. Bei größeren Bibliotheken können die nötigen Änderungen jedoch einen erheblichen Umfang annehmen. Diese Änderungen sollten, sofern möglich, in die Originalquellen eingepflegt werden (man spricht von upstream Änderungen), da man ansonsten bei neuen Ausgaben der Bibliothek die Anpassungen teilweise bis vollständig wiederholen muss.

6.3.3 Kommunikation nebenläufiger Prozesse

Als externe Bibliothek stellt sich ebenfalls die Kommunikationsbibliothek MPI (Message Passing Interface) dar. Dies ist ein verbreiteter Standard für Bibliotheken zur Distributed-Memory Parallelisierung. Verschiedene Bibliotheken implementieren diesen Standard, verwenden jedoch selbst zur Netzwerkkommunikation Betriebssystemfunktionen und zum Teil Hardware-nahe Programmierung. Eine algorithmische Differenzierung ist daher, trotz verfügbarem Quellcode, nicht möglich.

Die adjungierten Operationen lassen sich jedoch hier gut auf Basis der primalen Operationen ausdrücken [139]. Daher existieren Bibliotheken, welche die Funktionen des MPI-Standards auch für differenzierende Datentypen implementieren. Im Rahmen der Entwicklung wurden drei adjungierte MPI-Bibliotheken aufgrund technischer Belange in der Zusammenarbeit mit den verschiedenen AD-Werkzeugen angewandt: adjointMPI [140], adjoinableMPI [141] und MediPack[2]. Beispiele und Zeitmessungen im weiteren Verlauf basieren auf der Kombination CodiPack/MediPack.

Adjungierte MPI-Bibliotheken operieren als Zwischenschicht zwischen dem parallelen Programm und der eigentlichen MPI-Bibliothek. Im primalen Modus reichen sie alle Aufrufe unverändert weiter. Für den tangentialen Vorwärtsmodus reicht es, einfach die tangentialen Ableitungswerte mit zu kommunizieren. Für die Rückwärtsdifferentiation muss bei der Aufzeichnung des primalen Laufs vermerkt werden, welche Variablen zwischen den Prozessen kommuniziert wurden und welches ihre Bandeinträge auf den jeweiligen Prozessen waren. Bei der Rückwärtsauswertung des Bands müssen dann die adjungierten Werte in umgekehrter Richtung kommuniziert werden. Das Hochleistungs-Rechnen (High-Performance Computing HPC) stellt besondere Anforderungen durch die Verwendung asynchroner Kommunikation und kollektiver Operationen. Bei letzteren wird

[2] Entwicklung des Lehrstuhls SciComp der TU-Kaiserslautern

beispielsweise ein primaler Broadcast zu einer Reduktion mit Additionsoperator, wobei die entsprechende Behandlung der Bandeinträge und Generierung der Ableitungen der Ausdrücke eine erheblichen Implementierungsaufwand bedeuten kann.

Die adjungierbare MPI-Bibliothek kann dies automatisch erledigen, solange ihr bekannt ist, in welcher Kommunikation an welcher Stelle Gleitkomma-Werte übergeben werden. Dies bedingt, dass dem MPI-Aufruf die Datentypen der übergebenen Daten bekannt ist. Häufig werden jedoch zusammengesetzte Datentypen der MPI-Bibliothek als untypisierten Binärdatenblock versandt. Dies stellt für den primalen Löser keine Schwierigkeit dar, da der Datentyp auf Empfängerseite bekannt ist.

In einem solchen Fall muss die Kommunikation einer Variable der Bibliothek manuell mitgeteilt werden. Diese Aufgabe wird in TRACE durch automatisch generierten Code unterstützt, welcher für alle selbstdefinierten Datentypen die Position der enthaltenen Gleitkommavariablen extrahieren kann. Die Möglichkeit Gleitkommavariablen in untypisierten Binärdatenblöcken aufzubewahren oder zu transportieren ist eine der zentralen Herausforderungen der Umsetzung algorithmischer Differentiation mit C++ in realen Anwendungen. Diese sollten bei der Vorbereitung für algorithmische Differentiation in typensichere Konstrukte überführt werden. Kann dies nicht umgesetzt werden müssen die Indentität einer Variable und deren Ableitungsinformation beim Befüllen und Auslesen eines solchen Blocks dem AD-Werkzeug mitgeteilt werden.

6.3.4 Post-Processing und Seeding

Die Aufgabe des Strömungslösers besteht in der Abbildung des Netzes und der Randbedingungen auf einen räumlichen Strömungszustand. Für die Optimierung benötigt man jedoch skalare Bewertungsgrößen, die im Nachgang, im Post-Processing $J(q, x)$, berechnet werden. Das Post-Processing ist hier als eigenes Programm realisiert, welches die Strömungslösung einliest und daraus relevante Auswertegrößen berechnet. Das adjungierte Pendant wurde hier durch algorithmische Differentiation im Rückwärtstmodus erzeugt.

Da der Post-Processor ebenfalls parallelisiert ist, mussten die gleichen Arbeiten zur Differenzierung der MPI-Kommunikation auch hier vorgenommen werden. Das Post-Processing nutzt keine iterativen Lösungsverfahren, daher mussten keine Fixpunkt-Iterationsvorschrift verwendet werden. Laufzeit und Speicherverbrauch des Post-Processings stationärer Simulationen ist gering verglichen mit dem Simulationsverfahren selbst, daher mussten auch keine weiteren Optimierungstechniken eingebracht werden.

Für die adjungierte Differenzierung der Strömungssimulation benötigt man die Richtungsableitungen des Post-Processings bezüglich q und x.

Hierzu werden q und x als aktiv markiert und der primale Durchlauf des Post-Processings aufgezeichnet. Anschließend wird die i-te Ausgabegröße nach Benutzervorgabe als zu differenzierende Größe gewählt, indem der Vektor $\bar{J}$ mit dem i-ten Einheitsvektor belegt wird. Danach wird das Band im Rückwärtsmodus durchlaufen, wonach die adjungierten Werte des Strömungszustands $\bar{q}$ und der Netzkoordinaten $\bar{x}$ feststehen. Diese werden in Ausgabedateien geschrieben, um sie im adjungierten Strömungslöser einzulesen.

Der Post-Processor für TRACE nutzt für geometrische Operationen, wie Verschneidungsoperationen und Interpolationen, auf dem Rechennetz die Visualization-Toolkit

(VTK) Bibliothek [142]. Dies ermöglicht, unter anderem Berechnungen auf frei definierbaren Schnittflächen durch das Rechennetz. Da diese Bibliothek nicht in templatisierter Form vorliegt, wurde eine spezifische Version mittels händischen Einfügens der AD-Datentypen differenziert. Dies erzeugt natürlich das Problem, dass neuere Versionen der Bibliothek erneut modifiziert werden müssten. Alternativ steht die Möglichkeit zur Verfügung, die Ableitungen nach Ergebnissen von VTK zu ignorieren, was jedoch die Konsistenz der Sensitivitäten auf solchen Auswerteflächen beeinflusst.

Zu Testzwecken sind einfache Bilanz-Funktionale zwischen Ein- und Austritt des Rechengebietes auch direkt im Strömungslöser integriert, so dass für diese Fälle der Lauf des adjungierten Post-Processors nicht durchgeführt werden muss.

6.3.5 Fixpunkt-Rückwärts-Iteration

Mit der auskonvergierten primalen Lösung q^N und dem Ergebnis des adjungierten Post-Processings $\bar{q}$ kann die Fixpunkt-Rückwärts-Iteration durchgeführt werden.

Zur Durchführung der adjungierten Fixpunt-Iterationsvorschrift Gl. 4.36 mit AD Techniken ist dann

$$\bar{q}^{(0)} = \left(\frac{\partial J}{\partial q}\right)^T \bar{J} \tag{6.10}$$

$$\bar{q}^{m+1} = \left(\frac{\partial J}{\partial q}\right)^T \bar{J} + \left(\frac{\partial G}{\partial q}\right)^T \bar{q}^m \quad m = 1..M \tag{6.11}$$

Die Gleichungen 6.10 wird durch den algorithmisch differentierten Post-Processor berechnet.

Zur Iteration des adjungierten Zustands (Gl. 6.11) wird der Strömungszustand nach der letzten durchgeführten primalen Berechnung $q^{(N)}$ wiederhergestellt. Darauf werden die Zustandsvariablen q als aktiv markiert, G ausgeführt und die durchgeführten Operationen auf Band aufgezeichnet. Anschließend wird iterativ $\bar{q}$ rückwärts durch das Band transportiert und zum differenzierten Funktional addiert.

Ein entscheidender Punkt für den beschriebenen Algorithmus ist dabei die Definition des Zustandsvektors q. Betrachtet man allein die diskreten Navier-Stokes Gleichungen 3.10 so genügen zur Beschreibung des Zustands die konservativen Variablen $q_c = (\rho, \rho U, \rho V, \rho W, \rho E)^T$ in allen Netzzellen. Für den hier betrachteten Strömungslöser, und vermutlich für einen Großteil existierender Strömungslöser, reichen diese Variablen nicht zur Zustandsbeschreibung. Der Grund hierfür liegt in der Berechnungsreihenfolge abgeleiteter Größen in G und der Initialisierungsphase des Strömungslösers. In Abbildung 6.3 ist zu sehen, dass in der Initialisierung (G_I) abgeleitete Größen q_d berechnet werden, welche in der primalen Iteration zunächst benutzt ($G_{1/2}$), und erst später neu gerechnet werden ($G_{2/2}$). Definierte man $q = q_c$, würden bei einer Wiederherstellung des Zustandes die Werte von q_d nicht überschrieben und wären somit beliebig. Außerdem würde bei der Differenzierung von G ein Teil der Variablen nicht als Eingangsgrößen markiert und auch die Rückwärtsiteration wäre falsch, da nur die adjungierten Größen $\bar{q}_c$ iteriert würden.

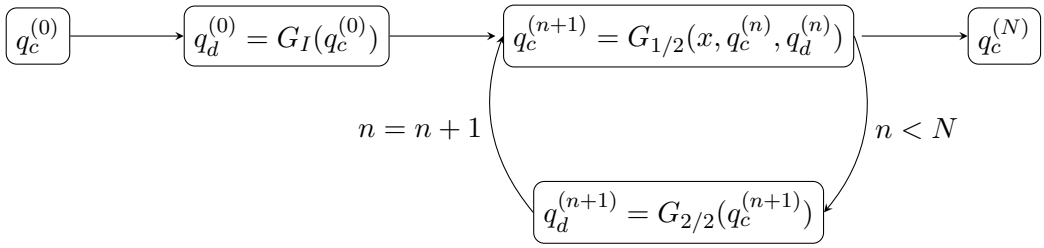


Abbildung 6.3: Schematische Berechnungsreihenfolge abgeleiteter Zustandsgrößen in der primalen Iteration

Zur Lösung des Problems könnte man die Reihenfolge der Schritte in G neu anordnen. In der realen Implementierung sind $G_I, G_{1/2}, G_{2/2}$ jedoch keine abgeschlossenen Prozeduren sondern Berechnungen welche an verschiedenen Stellen innerhalb der Initialisierungsphase stattfinden, bzw. eng in der primalen Iterationsvorschrift miteinander verbunden sind.

Daher wird $q = (q_c \quad q_d)$ als Zustand für die Rückwärtsiteration verwendet. Ob die Definition vollständig ist, kann durch wiederholte Durchführung einer G-Iteration und anschließender Wiederherstellung des vorhergehenden Zustands überprüft werden. Nur wenn eine vollständige Definition von q erfolgt ist, werden die entstehenden Zustände exakt gleich sein.

Im hier betrachteten Strömungslöser gehören zu q_d folgende Variablen:

1. Die Zustandsvariablen der Turbulenz- und Transitionsmodelle
2. Zustandsgradienten des unstrukturierten Lösers
3. Band-gemittelte Zustände auf Interface-Ebenen
4. Randbedingungs-Zustände bei relaxierten Randbedingungen

Die Differenz der adjungierten Werte des konservativen Zustands

$$\bar{R}_1 = \|\bar{q}_c^{m-1} - \bar{q}_c^{m}\|_1 \tag{6.12}$$

dient als adjungiertes Residuum zur Kontrolle der Konvergenz der Rückwärts-Iteration.

6.3.6 Abgeleitetet Geometriegrößen

Für die Berechnung der Ableitung des Funktionals nach Netzkoordinaten (s. Gl. 4.37)

$$\frac{\mathrm{d}\, J}{\mathrm{d}\, x} = \left(\frac{\partial J}{\partial x}\right)^T \bar{J} + \left(\frac{\partial G}{\partial x}\right)^T \mu$$

muss, unter anderem, $(\partial G/\partial x)^T \mu$ mittels AD ausgewertet werden. Grundsätzlich muss dies nur nach Konvergenz der Rückwärts-Iteration durchgeführt werden, praktisch kann

dies auch zur Ausgabe eines Zwischenergebnisses während der adjungierten Iteration erfolgen.

Hierzu würde man die Koordinaten des Rechennetzes x als Eingabevariablen markieren, eine G Iteration aufzeichnen und anschließend den adjungierten Lösungsvektor μ durch das Band transportieren.

Die tatsächlich implementierte Iterationsvorschrift des primalen Lösers $q^{n+1} = G(q^n, x)$, hängt jedoch nur indirekt von den Koordinaten des Rechennetzes x ab. In der Praxis verwendet man abgeleitete Größen der Netzkoordinaten $\xi = \xi(x)$, wie Zellvolumina, Flächen, Radien, Normalenvektoren, Projektionen in krummlinige Koordinatensysteme, Interpolationsgewichte insbesondere auch für nicht-kongruente Schnittstellen (nonconforming interfaces) und nicht-äquidistante schnelle Fouriertransformationen.

$$q^{n+1} = G(q^n, \xi(x)). \tag{6.13}$$

Diese abgeleiteten Größen ξ sind konstant in Rechnungen mit unbewegten Netzen[3]. Daher findet ihre Berechnung nicht in der Iterationsschleife G selbst statt. Sie werden nur einmalig berechnet, während der Initialisierung des primalen Lösers. Dieses Vorgehen kann man wie folgt beschreiben:

$$\begin{aligned} \xi_0 &= \xi(x) \\ q^{n+1} &= G(q^n, \xi_0) \end{aligned} \tag{6.14}$$

Bei rein mechanischer Anwendung von AD, auf die Iterationsvorschrift G erscheinen die Werte ξ_0 fälschlicherweise als konstant bezüglich x. Das AD-Werkzeug würde $\partial G/\partial x = 0$ berechnen.

Zur korrekten Berechnung muss $\xi(x)$ bei der Differentiation von G berücksichtigt werden. Die Ableitung der Iteration nach den Netzkoordinaten ist gemäß Kettenregel

$$\frac{\mathrm{d}\, G}{\mathrm{d}\, x} = \frac{\partial G}{\partial \xi}\frac{\partial \xi}{\partial x}. \tag{6.15}$$

Man könnte nun $\partial\xi/\partial x$ mittels AD berechnen. Aufgrund der Komplexität der Geometrieberechnung ist die Berechnung $\xi(x)$ nicht als eine Prozedur implementiert, sondern sie ist in viele kleine Routinen aufgeteilt, welche an verschiedenen Stellen der Initialisierung aufgerufen werden. Es ergeben sich die bereits im vorhergehenden Abschnitt geschilderten Probleme bei der Aufzeichnung der Initialisierung.

Eine Lösung ist die manuelle Extraktion der hier als $\xi(x)$ bezeichneten Berechnungen in eine eigene Routine P_ξ. Für die Differentiation mittels AD wird dann P_ξ und anschließend G hintereinander auf dem gleichen Band aufgezeichnet, die Geometrievariablen x als Eingabe- und die Zustandsvariablen q als Ausgabe markiert und $\bar{q}$ rückwärts durch das Band transportiert. Es wird also

$$\bar{x} = \left(\frac{\partial G}{\partial \xi}\frac{\partial \xi}{\partial x}\right)^T \bar{q} \tag{6.16}$$

[3]Diese sind nicht zu verwechseln mit Berechnungen im rotierenden Relativsystem. Rechnungen mit bewegten Netzen tauchen beispielsweise in der Aeroelastik auf, bei der sich feste Wände während der Rechnung verformen.

berechnet.

Die manuelle Extraktion von P_ξ für Adjungierungszwecke birgt die gleichen Konsistenzprobleme wie die, bereits diskutierte, manuelle Implementierung adjungierter Routinen. Abhilfe schafft hier die Möglichkeit, die gesamte Initialisierungsphase des Lösers aufzuzeichnen. Diese Aufzeichnung muss neben dem Band von G aufbewahrt werden, so dass bei Berechnung der Ableitungen nach x zunächst mit dem G-Band

$$\bar{\xi} = \frac{\partial G}{\partial \xi} \bar{q} \tag{6.17}$$

und anschließend

$$\bar{x} = \frac{\partial \xi}{\partial x} \bar{\xi} \tag{6.18}$$

berechnet wird.

Dieses Vorgehen verhindert eine inkonsistente Differenzierung von ξ. Es setzt jedoch noch immer voraus, dass die abgeleiteten Größen ξ vollständig bekannt sind, da sie als Ausgabe- respektive Eingabevariablen der Bandauswertung markiert werden müssen. Ferner bedeutet diese Vorgehensweise einen zusätzlichen Speicheraufwand, da die gesamte Initialisierungsphase und nicht nur ξ aufgezeichnet werden muss.

Es ist daher sinnvoll mit einer manuell extrahierten Variante von ξ zu arbeiten und diese regelmäßig mittels Aufzeichnung der Initialisierungsphase zu validieren.

6.3.7 Konstanter Präkonditionierer

Eine erhebliche Speicher- und Laufzeitersparnis des Lösers konnte durch Deaktivierung der Berechnung der linken Seite des impliziten Pseudo-Zeitschrittverfahrens erreicht werden.

Die Iterationsvorschrift des primalen Lösers nutzt zur Bestimmung der Zustandsänderung

$$\Delta q^n = q^{n+1} - q^n \tag{6.19}$$

das implizite Pseudozeitschrittverfahren (s. Gl. 3.12)

$$\left(\Delta \boldsymbol{\tau}^{-1} + \left. \frac{\partial \widetilde{R}}{\partial q} \right|_{q^{(n)}} \right) \Delta q^{(n)} = R^{(n)}.$$

Dieses kann man als präkonditionierte Fixpunktiteration auffassen

$$q^{(n+1)} = G(q^{(n)}, x) = q^{(n)} + P^{(n)}(q^{(n)}, x) R^{(n)}(q^{(n)}, x) \tag{6.20}$$

mit

$$P^{(n)}(q^{(n)}, x) = \left(\Delta \boldsymbol{\tau}^{-1} + \left. \frac{\partial \widetilde{R}}{\partial q} \right|_{q^{(n)}} \right)^{-1} \tag{6.21}$$

Für die adjungierte Iterationsvorschrift

$$\lambda^{k+1} = \left(\frac{\partial J(q_*, x)}{\partial q}\right)^T + \left(\frac{\partial G(q_*, x)}{\partial q}\right)^T \lambda_k \tag{6.22}$$

wird $G_q^T \lambda_k$ im AD-Rückwärtsmodus differenziert. Das Aufstellen der Matrix P zum jeweiligen Strömungszustand stellt einen wesentlichen Teil der Speichergröße für das Band bei der Aufzeichnung von G dar. Es kann beobachtet werden, dass die Neuberechnung von P im primalen Löser nahe eines Fixpunkts deaktiviert werden kann, ohne die Konvergenz zu beeinflussen. Da wir einen Fixpunkt q_* mit $R(q_*) = 0$ angenommen haben, kann man in der adjungierten Iteration

$$\left(\frac{\partial G}{\partial q}\right) = 1 - (\frac{\partial P}{\partial q} R + P \frac{\partial R}{\partial q}) \tag{6.23}$$

zu

$$\frac{\partial G}{\partial q} = 1 - P \frac{\partial R}{\partial q} \tag{6.24}$$

vereinfachen. Ein weiteres Argument hierfür ist die Konvergenz des Präkonditionierers selbst, so dass $\partial P / \partial q|_{q^*}$ verschwindet. Beweise für die Gleichheit der Konvergenzraten für Tangent-Vorwärts- und Black-Box-Modus finden sich in einer Veröffentlichung von Griewank [143] respektive Abschnitt 15.4 im Buch [17] des selben Autors.

Technisch lässt sich die vereinfachte Iterationsvorschrift auf zwei Arten erhalten: Weglassen der Neuberechnung von P während der Aufzeichnung von G, oder alleinige Deaktivierung der Band-Aufzeichnung während der Neuberechnung von P.

Eine unvorsichtige Deaktivierung von Teilen des Lösers kann fehlerhafte Ableitungen nach sich ziehen [17]. Da dieses Verfahren optional implementiert ist, kann jederzeit gegen die vollständige Iterationsvorschrift verifizieren werden.

6.4 Bewertung der Ansätze

Die im vorherigen Abschnitt dargestellten Ansätze zur Erzeugung eines adjungierten Strömungslösers haben jeweils spezifische Vor- und Nachteile. Die Bewertung wird im Folgenden, anhand von bedeutsamen Aspekten für die Verwendbarkeit, vorgenommen. Anschließend wird aus den getroffenen Bewertungen ein Fazit gezogen.

6.4.1 Konsistenz

Ein adjungierter Löser ist konsistent zum einem primalen Löser, wenn die berechnete Ableitung, genau die Ableitung der primalen Funktion ist. Schreibt man für die gesamte primale Auswertung $J(x)$, dann sollen die Ableitungen $\mathrm{d}J/\mathrm{d}x$ berechnet werden. In der Praxis kommt es jedoch regelmäßig vor, dass stattdessen eine andere Funktion $\tilde{J}(x) \approx J(x)$ differenziert wird: Die berechnete Ableitung $\mathrm{d}\tilde{J}/\mathrm{d}x$ ist im Allgemeinen nicht die Ableitung von J.

In dieser abstrakten Sichtweise scheint es offensichtlich, dass Inkonsistenzen der Ableitungen unerwünscht sind. Die Gründe, weshalb Inkonsistenzen vorkommen, können in folgende Kategorien eingeteilt werden:

1. Durch bewusste Entscheidungen zur Vereinfachung der Entwicklung des adjungierten Lösers.

2. Durch Fehler bei der Entwicklung des adjungierten Lösers.

3. Durch Änderung des primalen Lösers ohne entsprechende Anpassung des adjungierten Lösers.

Zur Kategorie (1) gehören bereits besprochene Vereinfachungen wie die Vernachlässigung von Modellen, beispielsweise Turbulenz- und Transitionsmodellen. Betrachtet man die Modellvariablen des Turbulenzmodells als unabhängig von (kleinen) geometrischen Variationen, wird dies (bei Wirbelzähigkeitsmodellen) als Constant Eddy Viscosity (CEV) Annahme bezeichnet. Sie scheint für eine große Klasse von Auslegungsproblemen das Ergebnis einer Optimierung mit diesen Ableitungen nicht wesentlich zu beeinflussen (vgl. [19, 54, 55]). Jedoch gibt es auch Hinweise aus der Aeroelastik, dass im Fall von Stoß–Grenzschicht–Interaktion die Ableitungen des Turbulenzmodells erheblichen Einfluss auf eine zeitlinearisierte Simulation haben können [144, 145]. Aufgrund der Verwandtschaft zeitlinearisierter und adjungierter Simulationen liegt die Vermutung nahe, dass auch adjungierte Simulationen solcher Strömungen mit CEV Annahme falsche Ergebnisse liefern, jedoch solche Konfigurationen bisher nicht in Vergleichsstudien optimiert wurden, oder der Einfluss des Effekts auf das Optimierungsresultat gering ist.

Die Vernachlässigung von Teilen des primalen Lösers bei der Adjungierung muss für jedes Modell und jede Klasse von Testfällen validiert werden. Daher sollte ein Löser zur Verfügung stehen, in dem alle Modelle des primalen Lösers auch im adjungierten Löser verfügbar sind. Die Erfahrung bei der Umsetzung beider beschriebener Ansätze hat gezeigt, dass dies im Top–Down–Ansatz mit einem extrem hohen Entwicklungsaufwand verbunden wäre. Im Bottom–Up–Ansatz hingegen konnte dieser Zustand erreicht werden.

Einen Sonderfall stellen vereinfachende Annahmen bei der Adjungierung des Lösungsverfahrens dar. Die Funktion $J(x)$ ist nur eindeutig für einen auskonvergierten Strömungszustand $J(q^*, x)$, welcher mittels des iterativen Verfahrens $q^{n+1} = G(q^n)$ berechnet wird und, wenn überhaupt, nur für $n \to \infty$ existiert. Schon die Annahme eines q^* stellt genaugenommen eine inkonsistente Vereinfachung dar. Verwirft man diese Annahme, bleibt nur die teure Black-Box Differenzierung des gesamten Lösungsprozesses entsprechend Gl. 4.30. Diese ist jedoch für die hier anvisierten Optimierungsanwendungen zu teuer, so dass gradientenbasierte Optimierung ineffizient wird.

Akzeptiert man diese Annahme und geht davon aus, dass ebenfalls ein adjungiertes $\bar{q}^*$ gefunden wird, beeinflussen die Vereinfachungen bei der Adjungierung des Lösungsverfahrens nicht das Ergebnis, sondern lediglich das Konvergenzverhalten.

Beide vorgestellte Vorgehensweisen zur Implementierung eines adjungierten Lösers verwenden die oben dargestellten Vereinfachungen. Bezüglich Konsistenz des Lösungsverfahrens sind sie daher als gleichwertig zu betrachten.

Ein Vorteil des Bottom–Up–Vorgehens ist jedoch die Verfügbarkeit eines Black-Box differenzierten Verfahrens, mit der das Vereinfachte Iterationsverfahren verglichen werden kann.

Die beiden Ursachen für Inkonsistenzen (2) und (3) unterscheiden sich nicht in der Auswirkung, sondern lediglich durch den Zeitpunkt ihres Auftretens. In beiden Fällen wird eine andere Funktion als die gewünschte differenziert, ohne dass dies dem Entwickler des adjungierten Lösers bewusst ist. Fehler bei der Herleitung und Implementierung des adjungierten Verfahrens betreffen naturgemäß die manuelle Differenzierung von Routinen (vgl. Abs. 5). Prinzipiell sind natürlich auch Fehler bei der Umsetzung der anderen Differenzierungstechniken möglich. Beispielsweise könnte das Werkzeug zur algorithmischen Differentiation fehlerhaft sein. Oder die Einbindung der adjungierten Berechnungsverfahren im Code könnte inkonsistent zu den hergeleiteten Vorgehensweisen sein. Das AD–Werkzeug sowie die Einbindung der Verfahren können jedoch gut getestet werden und sind auch deutlich weniger komplex als manuell differenzierte Routinen.

Inkonsistenzen durch eine Änderung einer primalen Routine, ohne dass die differenzierte Routine angepasst wurde, betrifft nur manuell differenzierte Routinen. Durch Regressionstests kann dies erkannt werden, jedoch muss jede differenzierte Routine dann getestet werden und bei Abweichungen wird zur Anpassung des adjungierten Lösers wieder manueller Aufwand fällig. Dies ist besonders bedeutend, da an einem Strömungslöser typischerweise mehrere Entwickler beteiligt sind. Vor allem werden mehr Entwickler primale Routinen ändern, als es Entwickler für die Konsistenzerhaltung der adjungierten Routinen gibt.

Ableitungen durch finite Differenzen oder AD durch Operator-Überladen, bleiben bei Änderung an den primalen Routinen konsistent. Bei AD durch Quelltext-Transformation muss sichergestellt werden, dass diese automatisch bei jedem Übersetzten des Quelltexts erneut durchgeführt wird.

Das Risiko von Routinen, die nach Änderung inkonsistente Ableitungen zeigen, sollte bei aktiver Fortentwicklung einer Software auf ein Minimum beschränkt werden. Im Bottom-Up Ansatz ist dies gegeben. Im Top-Down Ansatz kann der Anteil von manuell differenziertem Code durch den gezielten Einsatz von AD gering gehalten werden.

6.4.2 Testbarkeit

Das bewährteste Werkzeug zum Verifizieren adjungierter Löser sind finite Differenzen: Sie können nicht-intrusiv auf den gesamten Löser angewandt werden und sind einfach zu implementieren. Aufgrund der bereits diskutierten Schrittweitenproblematik erforden finite Differenzen jedoch manuelle Anpassung und sie bieten keine Sicherheit bezüglich ihres eigenen Fehlers. Daher können finite Differenzen nur als Richtwert dienen um, größere Abweichungen aufdecken zu können.

Genauere Verfahren, wie finite Differenzen komplexer Schrittweite und der tangentiale Vorwärtsmodus erfordern Eingriffe in den Quelltext. Hierin besteht ein wesentlicher Vorteil der Bottom-Up-Vorgehensweise: Die nötigen Eingriffe für intrusive Validierungsverfahren werden als Teil des Entwicklungsprozesses zwingend durchgeführt. Durch die vollständige Überladung aller Operationen kann der tangentiale Vorwärtsmodus der AD

leicht umgesetzt werden. Dieser Modus ist in AD-Werkzeugen verhältnismäßig leicht zu implementieren und zu testen[4], es ist nur ein Minimum an Implementierungsaufwand zur Verwendung nötig und er ist von keiner Schrittweitenproblematik betroffen. Daher wird der Vorwärtsmodus in dieser Arbeit als Verifikationswerkzeug eingesetzt. Zur Verifikation für viele Parameter kann beim Bottom-Up-Vorgehen der Black-Box-Rückwärtsmodus eingesetzt werden, was jedoch mit einem erheblichen Ressourcenaufwand verbunden ist.

Die AD-basierten Techniken setzen voraus, dass wirklich alle Anweisungen differenziert wurden - was insbesondere beim Einsatz von externen Bibliotheken eine Herausforderung darstellt. Dies kann durch den Vergleich mit finiten Differenzen in den Regel erkannt werden.

Die genannten Validierungen können natürlich ebenso genutzt werden, um einen Top-Down erstellten Löser zu validieren. Sie bedeutet dort jedoch zusätzlichen Aufwand für die Anwendung algorithmischer Differentiation auf den gesamten Löser.

6.4.3 Stabilität

Die wichtigste Hürde im praktischen Einsatz adjungierter Verfahren in Optimierungen ist die Stabilität des adjungierten Lösungsverfahrens. Da der diskret adjungierte Löser die Konvergenzeigenschaften des primalen Verfahrens erbt, konvergiert ein diskret adjungierter Löser für eine auf Maschinengenauigkeit auskonvergierte primale Lösung $|R(q^*)| < \varepsilon_m \ll 1$. Dies ist jedoch nur ein hinreichendes Kriterium. Konvergiert ein Verfahren in einen kontraktiven Fixpunkt, dann wird auch in der Nähe des Fixpunktes bereits adjungierte Konvergenz erreicht. In realen Anwendungsfällen wird eine perfekte Konvergenz auf Maschinengenauigkeit nur selten erreicht. Zum einen werden die primalen Rechnungen aus Gründen der Rechenressourcen deutlich vor Erreichen der Maschinengenauigkeit abgebrochen, zum anderen geraten Simulationen in Grenzzyklusschwingungen, durch welche das Residuum nicht weiter sinkt.

Grenzzyklusschwingungen können zum einen physikalischer Natur sein, beispielsweise durch periodisch abschwimmende Wirbel, oszillierende Stöße, oder instabile Ablöseblasen. Zum anderen, können sie jedoch auch numerischer Natur sein, beispielsweise durch periodisches Schwingen des Lösungsverfahrens aufgrund einer zu groß gewählten Schrittweite.

Streng genommen stellen solche Resultate mit Grenzzyklusschwingungen nicht die geforderte stationäre Lösung der zugrunde liegenden Gleichungen dar. Für kleinere physikalisch bedingte Schwingungen, oder bei vielen numerisch ausgelösten Grenzzyklen sind Integrale Auswertegrößen jedoch oft hinreichend genau konvergiert, so dass diese Werte verwendet werden können.

Linearisiert man nun um eine Lösung $q^{(n)}$ mit $R(q^{(n)}) \gg \varepsilon_m$ kann es sein, dass das adjungierte Lösungsverfahren instabil wird. Dies führt dazu, dass zu praktisch nutzbaren primalen Lösungen gegebenenfalls keine adjungierte Lösung gefunden werden kann. Dieses Problem ist in der Literatur zu linearisierten und adjungierten Lösern wohlbekannt [146] und es existiert eine Reihe von Vorschlägen zu ihrer Behebung. Giles und

[4] Die Seite autodiff.org weißt aktuell eine Zahl von 29 AD-Werkzeugen für die Sprachen C/C++ aus welche alle den Vorwärtsmodus unterstützen

Campobasso schlagen die Stabilisierung mittels der Reverse-Projection-Method (RPM) vor [147]. Im Kern der RPM-Methode steht die Idee, dass nur wenige Eigenwerte von $\partial G/\partial q$ die Stabilitätsbedingung verletzen und diese sich immer noch nahe des Einheitskreises befinden. Das Lösungsverfahren wird anhand einer Approximation der größten Eigenwerte auf zwei Räume projeziert, einen stabilen und einen instabilen. Diese können dann mit unterschiedlichen Lösungsverfahren, beispielsweise einer Newton-Iteration für die instabilen Moden, behandelt werden. Das Verfahren wurde auch von Dwight und Brezillon [148] angewandt und wurde durch Albring et al. im Kontext des hier vorgestellten Lösers untersucht [149]. Einige Autoren argumentieren, dass das präkonditionierte GMRES-Verfahren eine ähnliche Stabilisierungseigenschaft besitzt, wie das RPM-Verfahren

Ein alternativer Ansatz besteht in der Behandlung des primalen Verfahrens. Da die schlechte Konvergenz des adjungierten Lösers durch fehlende Konvergenz des primalen Lösers verursacht wird, schlagen Xu et al. daher ein Lösungsverfahren für den primalen Löser vor, welches auch bei milden Instabilitäten in einen Punkt konvergiert [150].

Grundsätzlich kann eine stationäre Simulation mit einer Grenzzyklusschwingung auch als instationäre Simulation aufgefasst werden, welche nicht-zeitgenau, nämlich mittels des Pseudozeitschrittverfahrens durchgeführt wird. Dann müsste diese Simulation auch wie eine instationäre Simulation adjungiert werden, was mit der bereits oben genannten Black-Box Iterationsvorschrift möglich ist. Wie Krakos et al. zeigen, gibt es Fälle in denen auch bei leichten instationären Effekten nur die Black-Box-Adjungierte der zeitgenauen instationären Simulation exakte Gradienten liefert [151], wobei Fensterung des Signals im Zeit- oder Frequenzbereich hilfreich sein kann [152]. In anderen Fällen ist die Instationarität von ausgeprägter chaotischer Natur, so dass die Black-Box-Ableitung stark von der Wahl des gewählten Zeitfensters abhängt. Für solche Fälle entwickelte Wang das Least-Square-Shadowing [153] zur Berechnung einer nützlichen Ableitung. Least-Square-Shadowing führt jedoch zu einem Randwertproblem in Raum und Zeit, weshalb der Berechnungsaufwand des Verfahrens extrem hoch ausfällt.

Von diesen Phänomenen sind beide vorgestellten Vorgehensweisen zur Erstellung eines adjungierten Lösers betroffen, da beide Verfahren eine stationäre Lösung voraussetzen. Die genannten Stabilisierungsverfahren lassen sich ebenfalls auf beide Verfahren anwenden. Ein Vorteil des Bottom-Up Verfahrens liegt jedoch darin, dass bei der Umsetzung die Black-Box-Iteration als Validierungswerkzeug zur Verfügung steht, um die Ergebnisse stabilisierter Iterationen zu überprüfen.

6.4.4 Ressourcenbedarf

Ein Vergleich des Ressourcenbedarfs beider Löser für die Gradientenberechnung des betrachteten Testfalls CRISPmulti findet sich im Anwendungsteil dieser Arbeit (Abschnitt 8.3.6). Auch wenn die konkreten Zahlen von der numerischen Konfiguration sowie den gewählten Einstellungen der Lösungsverfahren abhängen, so sind die Ergebnisse dennoch signifikant: Der direkte Vergleich der beiden hier betrachteten adjungierten Lösern zeigt, dass der mittels Top-Down Methode implementierte Löser bezüglich Speicherbedarf und Laufzeit effizienter ist, als der mittels Bottom-Up Verfahren erstellte Löser.

Weitere Geschwindigkeitsoptimierungen für den Bottom-Up adjungierten Löser werden in zukünftigen Forschungsprojekten untersucht.

6.4.5 Modellierungsumfang

Eine der größten Herausforderungen im Top-Down-Vorgehen, ist die Modellvielfalt moderner Simulationsverfahren. Aus der Zahl der Veröffentlichungen zur Adjungierung von Turbulenzmodellen (vgl. 1.3), kann man schließen, dass bereits die Integration eines Turbulenzmodells in den adjungierten Löser eine gewisse Herausforderung darstellt. Turbulenzmodelle sind jedoch nur der Anfang, da der Modellierungsgrad in der Simulation von Turbomaschinen stetig erhöht wird.

Wichtige Beispiele sind die zahlreiche Modifikationen bestehender Turbulenzmodelle, die Modellierung des laminar-turbulenten Umschlags, der Einfluss von Einblasungen und Absaugungen und nicht strukturiert anschließbare Nebengeometrien sowie die Berücksichtigung nicht-idealer Gase.

Die Erfahrung bei der manuellen Adjungierung von Turbulenzmodellen lässt vermuten, dass jedes dieser Modellierungsthemen einen erheblichen Entwicklungsaufwand zur Adjungierung im Top-Down-Vorgehen bedeuten würde. Im Bottom-Up-Ansatz sind diese Modelle nach der Implementierung im primalen Löser erheblich leichter adjungiert zur Verfügung zu stellen. Hierzu muss hauptsächlich beachtet werden, dass die Programmier-Richtlinien für das Operator-Überladen eingehalten wurden, ob sich der Umfang des Zustands geändert hat, oder ob neue abgeleitete Größen aus der Initialisierung hinzugekommen sind. Eine Fallstudie hierzu bietet die Arbeit von Engels-Putzka et al. [154], in der vorgestellt wird, welche Maßnahmen nötig waren, um den Bottom-Up-Adjungierten Löser um das Harmonic-Balance-Verfahren des primalen Lösers [91, 92] zu erweitern.

Wird eine differenzierbare Funktion nichtdifferenzierbar repräsentiert, beispielsweise durch tabellierte Werte, müssen in beiden Ansätzen ein differenzierbarer Ersatz geschaffen werden.

6.4.6 Zusammenfassender Vergleich der Vorgehensweisen

Durch die beiden beschriebenen Vorgehensmodelle, Top-Down und Bottom-Up Adjungierung, können nachgewiesenermaßen einsetzbare adjungierte Löser erstellt werden [62, 135]. Im direkten Vergleich der entstandenen Löser ist zu sehen, dass die Bottom-Up Methode vorteilhaft bezüglich Konsistenzerhaltung, Testbarkeit und Modellierungsumfang ist. Der mittels Top-Down Verfahren erstellte Löser zeigt einen geringeren Ressourcenbedarf, insbesondere bezüglich Laufzeit und Speicherbedarf. Die Wahl der Vorgehensweise hängt also von der Gewichtung dieser Gütekriterien ab. Da Korrektheit eines Programms vor dessen Geschwindigkeit stehen sollte[5], basieren die weiteren Untersuchungen in dieser Arbeit auf dem per Bottom-Up Ansatz erstellten adjungierten Löser.

[5]Donald Knuth prägte den Satz: "premature optimization is the root of all evil-[155]

7 Ersatzmodellbasierte Optimierung

7.1 Ersatzmodelle zur Optimierung

Im Kontext dieser Arbeit bedeutet der Begriff Modell ganz allgemein eine numerische Simulation eines physikalischen Systems welche als mathematische Funktion beschrieben werden kann. Insbesondere ist damit die Strömungssimulation einer konkreten Turbomaschinenkomponente gemeint, deren Form parametrisiert ist. Die Simulation bildet den Vektor der Entwurfsvariablen $x \in \mathcal{D} \subset \mathbb{R}^N$ aus dem Entwurfsraum $\mathcal{D}$ auf Zielfunktionswerte $f(x) \in \mathbb{R}^M$ ab.

Diese Zielfunktion möchte man minimieren, jedoch ist ihre Auswertung mit einem hohen Rechenaufwand verbunden, so dass die benötigte Zahl von Auswertungen des Simulationsmodells für eine Optimierung nicht praktikabel ist.

Eine Möglichkeit dennoch Optimierungen durchzuführen besteht darin, das Simulationsmodell durch ein vereinfachtes Modell $\widehat{f}(x)$ zu ersetzen und auf diesem nach einem Optimum zu suchen. Hierzu könnte man das physikalische Modell vereinfachen, beispielsweise durch Beschränkung auf zweidimensionale Effekte oder Verwendung der Euler-Gleichungen, korrelationsbasierte Ansätze etc. . Solche Vereinfachungen sind jedoch nur fallspezifisch möglich und müssen gerechtfertigt sein, um keine wesentlichen Effekte zu vernachlässigen. Die in dieser Arbeit verwendeten RANS Modelle sind bereits als optimaler Kompromiss zwischen Rechenzeit und Modellierungsgüte gewählt worden. Eine weitere Reduzierung des Modellierungsgrads ist nicht wünschenswert.

Ein anderes Verfahren ist die Ersatzmodellierung, bei welcher aus den bereits beobachteten Antworten (Ausgaben) des Simulationsmodells ein weiteres Modell gebildet wird. Dieses approximiert Zielfunktionswerte an unbekannten Stellen, basierend auf den bereits beobachteten Antworten des Simulationsmodells. Da solche Modelle dabei an Stelle des Simulationsmodells verwendet werden, heißen sie Ersatzmodelle oder Meta-Modelle. Synonym werden sie auch Antwortflächen genannt.

Wichtig ist, dass die Auswertung des Ersatzmodells erheblich günstiger ist, als die des Simulationsmodells. Optimierungsverfahren die sehr viele Zielfunktionsauswertungen benötigen, können dann das Ersatzmodell befragen und auf diesem ein (vorhergesagtes) Optimum suchen. Natürlich wird ein Ersatzmodell nicht für alle möglichen Punkte die gleichen Vorhersagen liefern wie das Simulationsmodell. Das Ersatzmodell wird immer einen Vorhersagefehler beinhalten, da das zu lernende System erheblich komplexer ist. Dieser Vorhersagefehler kann jedoch reduziert werden, indem weitere Beobachtungen des Modells in das Ersatzmodell integriert werden.

7.2 Radiale Basisfunktionen

Ein Ersatzmodell $\widehat{f}$ soll Werte einer Funktion in N unabhängigen Variablen $f : \mathbb{R}^N \mapsto \mathbb{R}$ vorhersagen, von der lediglich D einzelne Beobachtungen $(x_i, y_i = f(x_i))$ mit $i = 1, ..., D$ bekannt sind. Die verschiedenen Ausgabegrößen der Simulation $f(x) \in \mathbb{R}^M$ werden jeweils durch eigene Ersatzmodelle behandelt.

Eine bekannte Möglichkeit zur Ersatzmodellierung wären polynome mehrerer Variablen

$$\widehat{f}(x) = \sum_{|\alpha| \leq m} w_\alpha x^\alpha. \tag{7.1}$$

Für den multiindex $\alpha = (\alpha_1, \ldots, \alpha_N)$ gilt $|\alpha| = \sum_{j=1}^{N} \alpha_j$ sowie $x^\alpha = \Pi_{i=1}^{N} x_i^{\alpha_i}$. Diese haben jedoch Nachteile: Als Basisfunktion $\psi^{(i)}$ stehen alle möglichen Kombinationen aus Dimension und Potenz zur Wahl. Beispielsweise stehen bei drei Variablen und maximal quadratischen Polynomen schon 27 mögliche Basisfunktionen $\{x_1^p x_2^q x_3^r \forall p, q, r \in 0, 1, 2\}$ zur Auswahl. Nach der Wahl des maximalen Polynomgrads m können die Parameter w_i dann beispielsweise mittels kleinster Quadrate Methode bestimmt werden. Die Wahl des maximalen Polynomgrads m ist kritisch für die Flexibilität: Bei zu kleiner Wahl von m ist das Modell zu wenig flexibel, und kann den Verlauf der zu lernenden Funktion nicht wiedergeben. Bei zu großem m, kann es auch komplexere Funktionen wiedergeben, neigt aber bei ungünstiger Stützstellenwahl zu Überschwingern.

Die Anpassung eines mehrdimensionalen polynomialen Ersatzmodells ist im Allgemeinen ein kombinatorisches Problem, bei dem zunächst m, und dann n aus N^{m+1} möglichen Basisfunktionen für einen N-Dimensionalen Entwurfsraum gewählt und anschließend die n Gewichte gefunden werden. Das Problem polynomialer Ansätze ist also, dass die Ansatzfunktion und Stützstellenwahl passend zur Zielfunktion eingestellt werden muss.

Daher wird in der Regel ein anderer Ansatz gewählt, bei dem die Basisfunktionen keine Polynome sind, sondern radiale Basisfunktionen welche an einem Punkt im Raum zentriert sind, Funktionen des Radius von dort sind und mit steigendem Radius abklingen.

$$\widehat{f}(x) = \sum_{i=1}^{n} w_i \psi(\|x - c^{(i)}\|) \tag{7.2}$$

Beispiele für Basisfunktionen sind dabei

- invers Multi-Quadratisch $(1 + r^2)^{\beta/2}$, $\beta < 0$
- Gauß-Basis $\psi(r) = e^{-r^2/(2\sigma^2)}$

Werden die Basis-Zentren $c^{(i)}$ auf die gegebenen Punkte $c^{(i)} = x^{(i)}$ für $i = 1, ..., D$ gelegt, so erhält man zur Bestimmung ein lineares Gleichungssystem

$$\Psi w = y, \tag{7.3}$$

mit der Matrix $\Psi = \psi(\|x^{(i)} - x^{(j)}\|), i, j = 1, ..., D$. Ein Vorteil des Ansatzes ist, dass er zwar linear in den Gewichten w ist, jedoch aufgrund der Basisfunktionen ψ nichtlineare Funktionen wiedergeben kann.

Die Gewichte w können dann durch $w = \Psi^{-1}y$ berechnet werden, vorausgesetzt die Matrix ist invertierbar. Bei geeigneter Wahl der Basisfunktion, unter anderem der Gauß-Funktion, ist die Matrix positiv-definit. Das nun folgende Kriging kann als Erweiterung dieses Ansatzes aufgefasst werden.

7.3 Kriging

Die Ersatzmodellierungstechnik Kriging soll hier dargestellt werden, um ihre Anwendung im Rahmen gradientenbasierter Optimierungen zu diskutieren. Die folgende Beschreibung soll einen Überblick über das implementierte Verfahren geben, soweit es für die gradientenbasierte Optimierung relevant ist. Ausführliche Herleitungen finden sich bei Jones [156] und Sacks et al. [31]. Zudem existiert ein umfassendes Lehrbuch von Forrester et al. [38]. Die Auswahl der hier beschriebenen numerischen Verfahren entspricht den von Schmitz ausgewählten Methoden [73].

Man modelliert die beobachteten Antworten des Modells $y = [y^{(1)}, y^{(2)}, ..., y^{(D)}]$ an den Abtastpunkten $X = [x^{(1)}, x^{(2)}, ..., x^{(D)}]$ als Vektor von Zufallsvariablen, einem Zufallsfeld $\mathbf{Y} = [Y(x^{(1)}), ..., Y(x^{(D)})]^T$. Für die weitere Betrachtung nehmen wir an, dass die $x \in [0, 1]$ sind, was in der Praxis durch Skalierung mit geeigneten Ober- und Untergrenzen für die Entwurfsvariablen erreicht wird. Außerdem nehmen wir an, dass $\mathbf{Y}$ gaußverteilt und stationär ist und daher einen konstanten Erwartungswert $E[\mathbf{Y}] = \mu$ besitzt. Dies führt zum „Ordinary-Kriging“.

Es mag abwegig erscheinen ein Computermodell als zufälligen Prozess zu betrachten, schließlich sind die hier verwendeten Simulationsmodelle deterministisch. Es handelt sich hier jedoch um subjektiven Zufall, den Hampe [157] wie folgt definiert: „Sofern bei einem Ereignis davon ausgegangen werden kann, dass sein Auftreten Regeln unterliegt, die Beobachter diese Regeln jedoch nicht kennen oder es zu viele sind, um ihnen eine Prognose für das Auftreten der Ereignisse zu erlauben, handelt es sich um einen subjektiven Zufall.“

Da die Antwort des Simulationsmodells nicht genau vorhergesagt werden kann, ohne das Modell selbst auszuwerten, und dies nur im begrenzten Maße möglich ist, werden seine Antworten als subjektiv zufällig modelliert.

Für die Vorhersage der Funktion $f(x)$ an noch nicht beobachteten Punkten, nimmt man an, dass ihre Funktionswerte räumlich korreliert sind. Räumliche Korrelation bedeutet, dass die beobachteten Antworten nah beieinander liegende Punkte x sich ähnlicher sind als die weiter auseinanderliegender Punkte. Somit können aus bereits beobachteten Antworten statistische Abschätzungen unbekannter Antworten gewonnen werden. Dies erlaubt die Ermittlung einer Korrelationsfunktion für das Zufallsfeld $\mathbf{Y}$.

Man nimmt eine parametrisierte A-priori Verteilung an und versucht für die Parameter optimale Werte anhand der gegebenen Beobachtungen zu ermitteln.

In dieser Arbeit wird als Korrelationsfunktion die Gauß-Korrelationsfunktion verwendet. Diese Funktion ist zwar nur eine von vielen möglichen Wahlen, jedoch ist sie die bei weitem populärste Formulierung. Ihre Popularität geht maßgeblich auf die Veröffentlichung von Sacks et al. „Design and analysis of computer experiments“, kurz DACE [31] zurück. Die Korrelationsfunktion in der DACE Formulierung ist eine Funktion der Orte

zweier Punkte

$$\mathrm{Corr}(Y(x_i), Y(x_j)) := e^{-d(x_i, x_j)} \tag{7.4}$$

mit der inversen Korrelationslänge

$$d(x^{(i)}, x^{(j)}) := \sum_{k=1}^{N} \theta_k |x_k^{(i)} - x_k^{(j)}|^{p_k}. \tag{7.5}$$

Die Parameter p_k mit $0 < p_k \leq 2$ wirken auf die Glattheit des Ersatzmodells: $p_k = 2$ entspricht einer glatten, unendlich oft differenzierbaren Funktion, $p_k < 2$ einer nicht differenzierbaren fraktalen Funktion, wobei p_k die fraktale Dimension steuert. Mit $p_k = 2$ entspricht die Funktion der Korrelationsfunktion eines Gauß-Prozesses. Da die Differenzierbarkeit für die spätere Gradientenerweiterung benötigt wird, wird diese Formulierung in dieser Arbeit ausschließlich verwendet.

Sie ist verwandt mit der Gauß-Basisfunktion aus der Beschreibung radialer Basisfunktionen, jedoch besitzt die Kriging-Korrelationsfunktion einen Parameter für jede Dimension des Entwurfsraums, während die radiale Basisfunktion nur einen globalen Parameter besitzt. Das Kriging Modell ist somit allgemeiner als radiale Basisfunktionen mit Gauß-Basis.

Für gleiche Punkte $x_i = x_j$ ist die Korrelation 1, für weit voneinander entfernte Punkte $|x_i - x_j| \to \infty$ wird die Korrelation 0. Die Parameter der Funktion $\theta_k > 0$ mit $k \in 1, ..., N$, bestimmen dabei wie schnell die Korrelation mit steigendem Abstand in der k-ten Dimension abnimmt. Große Werte für θ_k bedeuten dabei, dass die zu modellierende Funktion sich in der k-ten Dimension schnell ändert: Nur sehr nah beieinander liegende Punkte korrelieren miteinander, für weiter entfernte Punkte ist die Korrelation der Funktionswerte niedrig. Man sagt die Variable $x^{(k)}$ ist aktiv. Für kleine Werte von θ_k spricht man von einer inaktiven Variable, denn trotz großer Abstände in $x^{(k)}$ gibt es dennoch eine starke Korrelation der zugehörigen Funktionswerte. Wegen der Wirkung der Parameter θ_k^{-p} heißen sie auch Korrelationslängen. Als Parameter des Meta-Modells werden die Parameter der Korrelationsfunktion auch Hyperparameter genannt.

Kriging ist eine Form des des überwachten Maschinenlernens [158]. Die Ermittlung des Kriging-Prädiktors wird als Training bezeichnet.

7.3.1 Training des Kriging Modells

Das Einstellen optimaler Hyperparameter θ_k, basiert auf dem Maximum-Likelihood Ansatz. Hierbei wird die Likelihood-Funktion maximiert, welche die Wahrscheinlichkeit beschreibt, dass die beobachteten Daten, unter den gegebenen Annahmen, durch das Modell produziert wurden.

Die Likelihood-Funktion beschreibt die Wahrscheinlichkeit, dass die beobachteten Daten Y durch das Modell $\hat{f}$ produziert wurden.

Mit der Annahme eines stationären Prozesses ist die Kovarianz-Matrix

$$\Psi = [\Psi]_{i,j} = \mathrm{Cov}(x^{(i)}, x^{(j)}) = \sigma^2 \mathrm{Corr}(x^{(i)}, x^{(j)}) \quad \text{für} \quad i, j = 0 \ldots D \tag{7.6}$$

und der Annahme der Stationarität kann die Likelihood-Funktion durch die beobachteten Daten ausgedrückt werden (vgl. [38]):

$$L = \frac{1}{(2\pi\sigma^2)^{D/2}|\Psi|^{1/2}} \exp\left[-\frac{(Y - F\mu)^T \Psi^{-1}(Y - F\mu)}{2\sigma^2}\right]. \tag{7.7}$$

Wobei $F = (1...1)$ Aus dieser Gleichung erhält man Maximum-Likelihood Schätzungen für den Erwartungswert und die Varianz (vgl. [38])

$$\hat{\mu} = \frac{F^T \Psi^{-1} Y}{F^T \Psi^{-1} F} \tag{7.8}$$

$$\hat{\sigma}^2 = \frac{1}{D}(Y - F\mu)^T \Psi^{-1}(Y - F\mu). \tag{7.9}$$

Die Schätzung der Varianz ist bei einer kleinen Datenbasis sehr empfindlich gegenüber kleinen Änderungen. Im Fall von Multi-Fidelity-Kriging ist eine analytische Berechnung der verschiedenen Varianzschätzer nicht möglich. Daher wird σ als zusätzlicher Hyperparameter während der Likelihood-Maximierung behandelt.

Durch Einsetzen dieser Schätzungen in die Likelihood-Funktion, Logarithmierung und Vernachlässigung aller Terme die bezüglich θ konstant sind, entsteht die konzentrierte log-Likelihood Funktion

$$\ell(\theta) \approx -\frac{1}{2}\ln(\det(\Psi)) - \frac{D}{2}\ln(\hat{\sigma}^2). \tag{7.10}$$

Durch Maximierung dieser Gleichung über θ wird das wahrscheinlichste Modell zu den Beobachtungen gefunden. Hierfür werden in der Literatur oft gradientenfreie Optimierungsverfahren vorgeschlagen. Im hier betrachteten Anwendungsfeld hat sich ein Gradientenabstiegsverfahren bewährt, deren Abstiegsrichtung dem „Resilient Backpropagation“-Verfahren aus dem Bereich neuronaler Netze [159] entlehnt ist:

$$\theta_k^{(t+1)} = \theta_k^{(t)} + \gamma^{(t)} \mathrm{sgn}\left(\left.\frac{\partial \ell}{\partial \theta_k}\right|_{\theta^{(k)}}\right) \tag{7.11}$$

Die hierfür benötigten partiellen Ableitungen der log-Likelihood Funktion sind:

$$\frac{\partial \ell}{\partial \theta_k} = -\mathrm{Tr}\left(\Psi^{-1}\frac{\partial \Psi}{\partial \theta_k}\right) + (Y - F\mu)^T \Psi^{-1}\frac{\partial \Psi}{\partial \theta_k}\Psi^{-1}(Y - F\mu). \tag{7.12}$$

Durch die Definition von

$$\theta_k = e^{\widetilde{\theta}_k} \tag{7.13}$$

und des Trainings von $\widetilde{\theta}_k$, kann die Restriktion $\theta_k > 0$ aus Gl. 7.5 bei der Optimierung vermieden und die Konditionierung der Trainingsoptimierung verbessert werden. Eine der zentralen Herausforderungen ist die effiziente numerische Implementierung des Trainigsalgorithmus [73].

7.3.2 Kriging Vorhersagen

Ein trainiertes Kriging Modell kann zur Vorhersage der Funktion f an beliebiger Stelle $\mathbf{x}$ verwendet werden.

Mit der Korrelation eines vorherzusagenden Punktes x zu allen bereits betrachteten Punkten x_i

$$\mathbf{c}(x) = (\mathrm{Cov}(x, x^{(i)})) \tag{7.14}$$

ist die Kriging-Vorhersage des Erwartungswerts

$$\widehat{y}(x) = \mu + c^T(x)\Psi^{-1}(Y - F\mu). \tag{7.15}$$

Sie entspricht der Annahme, dass der Mittelwert der Beobachtungen eine gute Näherung für die Vorhersage ist, welche im Bereich der beobachteten Daten korrigiert wird. Die Vorhersage ist der beste lineare erwartungswerttreue Schätzer („best linear unbiased predictor“ - BLUP [160]).

Die Unsicherheit der Vorhersage kann als Varianz der Vorhersage abgeleitet werden (vgl. [38]):

$$\hat{s}^2(x) = \left(\sigma^2 - c^T(x)\Psi^{-1}c(x) + \frac{(c^T(x)\Psi^{-1}F - \mu)^2}{F^T\Psi^{-1}F}\right). \tag{7.16}$$

An den Punkten, welche in das Training eingegangen sind, interpoliert das Modell die Funktionswerte exakt und die Vorhersage ist varianzfrei. Dies wird mit der Definition von $c(x) = \mathrm{Cov}(x, x_i)$ klar. Wertet man nun das Modell an einem der Trainingspunkte x_j aus, so entspricht $c(x_j)$ der j-ten Spalte der Kovarianzmatrix den wir hier mit Ψ_j bezeichnen. In Gleichung 7.15 ist wegen der Symmetrie von Ψ dann

$$c^T(x_j)\Psi^{-1} = \Psi_j^T\Psi^{-1} = (\Psi^{-1}\Psi_j)^T = e_j. \tag{7.17}$$

Und die Vorhersage wird zu

$$\widehat{y}(x_j) = \mu + e_j(Y - F\mu) = \mu + Y^{(j)} - \mu = Y^{(j)}. \tag{7.18}$$

In der Vorhersage der Varianz in Gleichung 7.16 ist mit ähnlicher Argumentation (vgl. [32])

$$\hat{s}^2(x) = 0. \tag{7.19}$$

7.3.3 Gradientenerweitertes Kriging

Die obenstehende Kriging Formulierung verwendet nur Funktionswerte der zu modellierenden Funktion Y. Im Kern dieser Arbeit steht die Integration von Gradienteninformation in den Optimierungsprozess. Daher wird ein gradientenbasiertes Kriging (Gradient Enhanced Kriging) verwendet. Gradienten der zu modellierenden Funktion $\partial Y/\partial x$ können auf zwei Arten in das Modell einbezogen werden: indirekt und direkt. Im indirekten Verfahren werden zu jedem Samplingpunkt x_i durch Extrapolation erster Ordnung weitere Punkte $x_i + \varepsilon e_j$ und Funktionswerte

$$\tilde{Y}(x_i + \varepsilon e_j) = Y(x) + \varepsilon e_j \frac{\partial Y}{\partial x_j} \tag{7.20}$$

hinzugefügt. Der Vorteil dieses Vorgehens liegt in der unmodifizierten Anwendung der Kriging Formulierung. Eine große Schwierigkeit stellt jedoch die Wahl des Schrittweitenparameters ε dar. Zu große Werte führen durch Vernachlässigung Terme höherer Ordnung zu fehlerhaften Sampling-Daten, zu kleine Werte führen zu einer schlecht konditionierten Korrelationsmatrix und somit zu Problemen im Training des Modells [40]. Daher wird hier der direkte Ansatz des Direct Gradient Enhanced Kriging (DGEK) verwendet, bei dem zusätzliche Korrelationen eingeführt werden. Neben den Korrelationen zwischen den Funktionswerten zweier Punkte werden nun auch Korrelationen zwischen Funktionswerten und Ableitungen und Korrelationen zwischen Ableitungen untereinander eingeführt. Die Kovarianz eines Zufallsfelds an einer Stelle x_i und den partiellen Ableitungen an anderer Stelle x_l ist die erste Ableitung der Kovarianzfunktion

$$\mathrm{Cov}(x_i, \frac{\partial Y(x_l)}{\partial x_l^n}) = \frac{\partial \mathrm{Cov}(x_i, x_l)}{\partial x_l^n}. \tag{7.21}$$

Die Korrelation partieller Ableitungen an verschiedenen Stellen entspricht der zweiten Ableitung der Kovarianzfunktion

$$\mathrm{Cov}(\frac{\partial Y}{\partial x_l^m}, \frac{\partial Y}{\partial x_l^n}) = \frac{\partial^2 \mathrm{Cov}(\mathbf{x}_j, \mathbf{x}_l)}{\partial \mathbf{x}_l^m \partial \mathbf{x}_l^n}. \tag{7.22}$$

Diese Einträge werden in der gradientenerweiterten Kovarianzmatrix angeordnet:

$$\widetilde{\mathbf{\Psi}} = \begin{pmatrix} \mathrm{Cov}(x_i, x_j) & \frac{\partial \mathrm{Cov}(\mathbf{x}_i, \mathbf{x}_l)}{\partial \mathbf{x}_l^n} \\ \underbrace{\frac{\partial \mathrm{Cov}(\mathbf{x}_l, \mathbf{x}_i)}{\partial \mathbf{x}_l^n}}_{\text{D Einträge}} & \underbrace{\frac{\partial^2 \mathrm{Cov}(\mathbf{x}_j, \mathbf{x}_l)}{\partial \mathbf{x}_l^m \partial \mathbf{x}_l^n}}_{P \text{ Einträge}} \end{pmatrix} \begin{matrix} \} \text{ D Einträge} \\ \} \text{ P Einträge} \end{matrix} \tag{7.23}$$

Die Indices $i, j, l \in 1, ..., D$ laufen über die bereits beobachteten Punkte, während $m, n \in 1, ..., N$ über die Dimension des Entwurfsraums laufen. P bezeichnet die Gesamtzahl der vorliegenden partiellen Ableitungen. Stehen alle partiellen Ableitungen an allen ausgewerteten Stellen zur Verfügung ist $P = D \cdot N$ und die Matrix hat daher die Dimension $D(1+N) \times D(1+N)$. Diese erweiterte Kovarianzmatrix $\widetilde{\mathbf{\Psi}}$ kann in die Gleichungen der Kriging Vorhersage Gl. 7.15 und ihrer Varianz Gl. 7.16 eingesetzt werden. Hierzu müssen einige Definitionen aus den Abschnitten 7.3.1 und 7.3.2 entsprechend angepasst werden:

$$\begin{aligned} \mathbf{Y_s}^T &= [y_1, ..., y_D, \frac{\partial y_1}{\partial x_1}, ..., \frac{\partial y_D}{\partial x_n}], \\ \mathbf{F}^T &= [\ \underbrace{1, ..., 1}_{\text{D Einträge}}\ ,\ \underbrace{0, ...0}_{\text{P Einträge}}\ ], \\ \mathbf{c}_i(\mathbf{x})^T &= [\mathrm{Cov}(\mathbf{x_1}, \mathbf{x}), ..., \mathrm{Cov}(\mathbf{x_D}, \mathbf{x}), \\ &\quad \frac{\partial \mathrm{Cov}(\mathbf{x_1}, \mathbf{x})}{\partial x_1^1}, ..., \frac{\partial \mathrm{Cov}(\mathbf{x_D}, \mathbf{x})}{\partial x_D^n}], \end{aligned}$$

Außerdem muss die Zahl der Beobachtungen D in allen anderen Kriging Gleichungen durch die Zahl der Beobachtungen und partiellen Ableitungen M ersetzt werden. Liegen an allen Beobachtungspunkten alle partiellen Ableitungen vor, gilt $M = D \cdot (1 + N)$.

7.4 Regularisierung der Korrelationsmatrix

Für das Training und die Vorhersage des Kriging wird in der log-Likelihood Funktion (s. Gl. 7.12) eine inverse Korrelationsmatrix Ψ^{-1} benötigt. Die Verwendung der Gauß Korrelationsfunktionen bedingt zwar, dass diese Matrix immer positiv definit ist, leider kann jedoch ihre Konditionszahl sehr groß werden, so dass die verwendete Cholesky Zerlegung fehlschlägt oder extrem fehlerbehaftet wird. Dies geschieht insbesondere dann, wenn die Dichte der Datenpunkte zu hoch wird.

Dass hierdurch eine schlecht konditionierte Korrelationsmatrix entsteht, kann leicht eingesehen werden wenn man sich den Verlauf der Korrelationsfunktion (s. Gl. 7.4) gegen die Extremfälle anschaut. Die Korrelationsmatrix besteht auf ihrer Hauptdiagonalen aus den Korrelationen der Datenpunkte mit sich selbst, welche per Definition $\mathrm{Corr}(x^{(i)}, x^{(i)}) = 1$ ist. Für den Fall großer, gewichteter Abstände

$$\sum_k \theta_k |x^{(i)} - x^{(j)}| \to \infty$$

gehen die Korrelationswerte abseits der Hauptdiagonalen gegen 0 (s. Gl. 7.4), die Matrix wird daher zur Einheitsmatrix. Werden die gewichteten Abstände hingegen klein

$$\sum_k \theta_k |x^{(i)} - x^{(j)}| \to 0$$

werden die Korrelationen 1 und die Matrix wird zur singulären Matrix in der alle Einträge 1 sind. Dies kann durch sehr dicht beieinander liegende Sampling-Punkte geschehen, oder durch kleine Werte für θ aus dem Training, welches eine starke Korrelation weit entfernter Punkte bedeutet. Abhängig von der geschätzten Korrelationslänge sollten also eine gewisse Dichte von Abtastpunkten nicht überschritten werden, um die Kondition des Kriging-Modells in Maßen zu halten. Eine tiefergehende Beschreibung dieser Argumentation kann bei Zimmermann [161] nachgelesen werden.

Dass die Korrelationsmatrix des Gradient Enhanced Kriging deutlich größer ist, als im gradientenfreien Kriging ist in der Literatur beschrieben worden (vgl. Zimmermann [40]). Dies kann an einem Beispiel veranschaulicht werden. Die Funktion

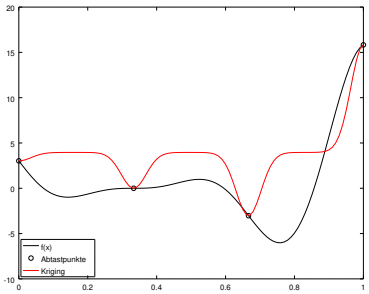
$$f(x) = (6x - 2)^2 \cdot \sin(12x - 4)$$

besitzt die Ableitung

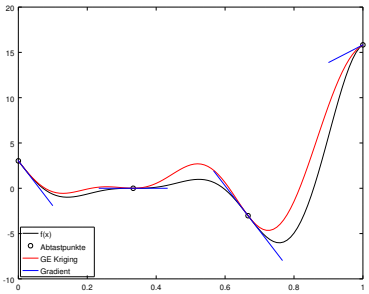
$$f'(x) = 12(6x - 2) \cdot \sin(12x - 4) + (6x - 2)^2 \cdot \cos(12x - 4) \cdot 12$$

In Abb. 7.1 sind gradientenfreie sowie gradientenbasierte Kriging-Modelle dieser Funktion und ihrer analytisch implementierter Ableitung dargestellt, wobei die Zahl der Trainigspunkte schrittweise gesteigert wurde.

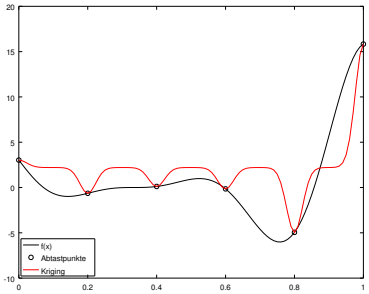
Betrachtet man den Verlauf der Konditionszahl von Ψ (s. Abb. 7.2), so sieht man, dass die Konditionszahl des gradientenbasierten Kriging Modells sehr schnell wächst, während sie im gradientenfreien Kriging auf moderate Werte beschränkt bleibt.



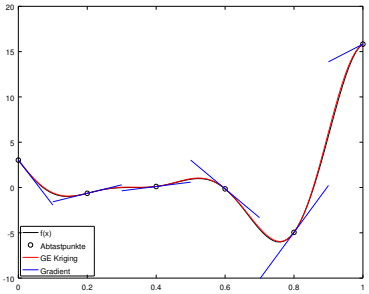
4 Abtastpunkte



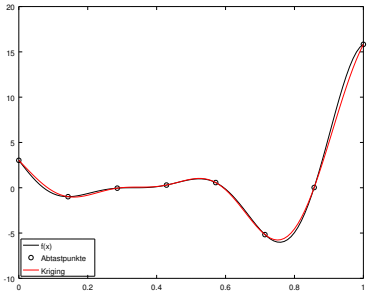
4 Abtastpunkte + Gradienten



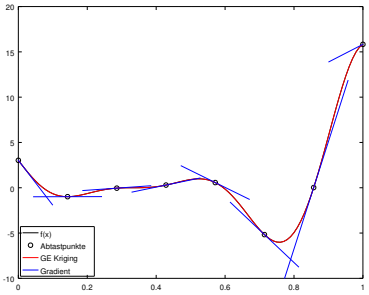
6 Abtastpunkte



6 Abtastpunkte + Gradienten



8 Abtastpunkte



8 Abtastpunkte + Gradienten

Abbildung 7.1: Steigerung der Informationsmenge für Kriging Modelle mit und ohne Gradienten für eine Beispielfunktion

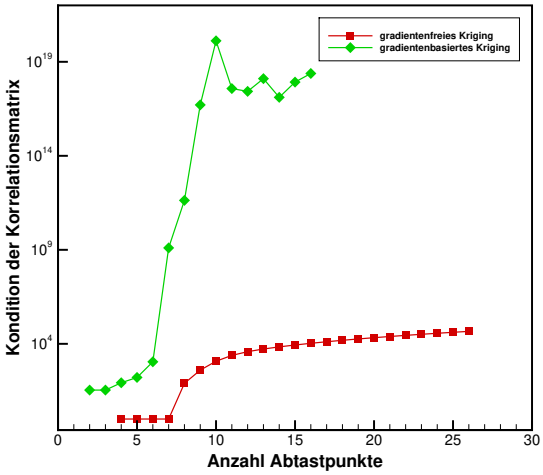


Abbildung 7.2: Studie der Konditionszahl über der Anzahl der Abtastpunkte

Seien $\lambda_i \neq 0, i = 1..,n$ die Eigenwerte der symmetrischen Kovarianzmatrix Ψ, dann ist ihre Konditionszahl

$$\kappa(\Psi) = \left| \frac{\max(\lambda_i)}{\min(\lambda_i)} \right|. \tag{7.24}$$

Darin sind $\max(\lambda_i)$ und $\min(\lambda_i)$ der größte, respektive kleinste Eigenwert der Kovarianzmatrix. Für eine schlecht konditionierte Matrix geht $\kappa(\Psi) \to \infty$.

Eine verbreitete Abhilfe stellt die Tikhonov-Regularisierung [162] durch Addition einer Regularisierungsmatrix Γ dar. Oft wählt man $\Gamma = \alpha \cdot E$ mit $\alpha \in \mathbb{R}$ und der Einheitsmatrix E. Sei λ ein Eigenwert von Ψ, dann ist der entsprechende Eigenwert der regularisierten Matrix $\lambda_r = \lambda + \alpha$.

Dies wird klar wenn man die regularisierte Matrix in die Eigenwertdefinition einsetzt:

$$\begin{aligned}
&(\Psi + \Gamma)\, v = \lambda_r v \\
\Leftrightarrow &\Psi v + \Gamma v = \lambda_r v \\
\Leftrightarrow &\Psi v = \lambda_r v - \alpha E v \\
\Leftrightarrow &\Psi v = (\lambda_r - \alpha)\, v \\
\Rightarrow &\lambda_r = \lambda + \alpha
\end{aligned}$$

Die Konditionszahl der regularisierten Matrix ist dann

$$\kappa(\Psi + \Gamma) = \left| \frac{\max(\lambda_i) + \alpha}{\min(\lambda_i) + \alpha} \right|. \tag{7.25}$$

Im Fall schlechter Konditionierung geht $\max(\lambda_i) \to 0$ und daher ist $\max(\lambda_i) + \alpha \approx \max(\lambda_i)$, für den kleinsten Eigenwert gilt $\min(\lambda_i) \to 0$ und daher $\min(\lambda_i) + \alpha \approx \alpha$:

$$\kappa(\Psi + \Gamma) \to \left| \frac{\max(\lambda_i)}{\alpha} + 1 \right|. \tag{7.26}$$

Durch Regularisierung verliert das Kriging Modell die Stetigkeit der Vorhersage an den zum Training vorgegebenen Punkten. Diese beruht darauf, dass c an den Trainingspunkten zu einer Spalte von Ψ wird. Durch die Addition eines Terms auf die Hauptdiagonale von Ψ verliert das Modell diese Eigenschaft.

Regularisiert man auf diese Weise die erweiterte Korrelationsmatrix $\widetilde{\Psi}$, so wird die Vorhersage der Funktionswerte genau so regularisiert wie die Vorhersage der Gradienten. Dies ist nicht wünschenswert, da die Gradienteninformation die Konditionszahl wesentlich stärker beeinflusst als die Funktionswerte. Die Gradienten unterliegen bei realistischen Prozessketten auch einem größeren Rauschen als die Funktionswerte (vgl. Abschnitt 8.3.2). Um den Einfluss der Gradienteninformation zu regularisieren würden gleichzeitig auch die Vorhersage der Funktionswerte an bekannten Stellen beeinflusst. Daher wird eine eigene Regularisierungsgröße für Gradienten verwendet:

$$\Gamma = \mathrm{diag}(\underbrace{\alpha \quad \alpha \quad \cdots \quad \alpha}_{D \text{ einträge}} \quad \underbrace{\beta \quad \beta \quad \cdots \quad \beta}_{P \text{ einträge}}). \tag{7.27}$$

Dieser Regularisierungsansatz enthält zwei Regularisierungs-Parameter: $\alpha > 0$ ist der Regularisierungs-Aufschlag für Funktionswert-Korrelationen, während $\beta > 0$ auf die Ableitungs-Korrelationen wirkt. Durch Steigerung von β kann die Gradienteninformation stärker regularisiert werden, um die Konditionszahl des Kriging Modells zu verbessern, ohne gleichzeitig die Vorhersage der bekannten Punkte weiter zu verschlechtern.

7.5 Generierung der Ausgangsdatenbasis

Um ein Kriging-Modell aufzubauen, muss ein initialer Satz von Beobachtungen des Simulationsmodells vorliegen. Die Wahl geeigneter Beobachtungspunkte wird in der Literatur als Versuchsplan (engl. Design of Experiments, kurz DOE) [31] bezeichnet. Ziel eines DOE ist es, einen Kompromiss aus einer möglichst guten Abdeckung des Entwurfraums und vertretbarem Rechenaufwand zu finden. Die anvisierten Entwurfsräume besitzen typischerweise 50-200 Dimensionen, so dass klassische, raumfüllende Verfahren wie Latin-Hypercube [163] in der Regel zu teuer sind. Für die vorliegenden Optimierungen kommt noch hinzu, dass möglicherweise Teile des Entwurfsraums unzulässige Regionen sind, da implizit formulierte Restriktionen verletzt seien können, oder Teile des Auswertungsprozesses abbrechen. Ferner liegt bei einer Optimierung in der Regel ein bereits bewerteter Ausgangsentwurf vor, in dessen Nähe auf jeden Fall nach Optima gesucht werden soll.

Um einen großen Satz erfolgreich bewertbarer Beobachtungspunkte zu erhalten, wird in dieser Arbeit die zufällige Variation des Ausgangsentwurfs als DOE-Verfahren verwendet. Jede Entwurfsvariable $x_i, i = 1 \ldots N$ wird mit einer Wahrscheinlichkeit p zur Mutation ausgewählt und falls sie gewählt wurde, im Intervall $[\underline{x}_i, \overline{x}_i]$ zufällig variiert. Hierbei kommt eine Gaussverteilung mit Mittelwert x_i zum Einsatz welche eine anzugegebende Varianz σ^2 erfüllt und bei $\underline{x}_i, \overline{x}_i$ abgeschnitten wird.

Durch die Wahl von p und σ^2 kann ein gradueller Kompromiss zwischen großer Entwurfsraumabdeckung auf der einen Seite und der Toleranz des Bewertungsprozess gegen zu ausgefallene Entwürfe auf der anderen Seite gefunden werden. Für kleine Werte der

Parameter werden Entwürfe favorisiert die in der Nähe des Ausgangsentwurfs liegen und somit eine höhere Wahrscheinlichkeit aufweisen, die Auswertungsprozesskette erfolgreich zu durchlaufen. Für große Werte wird eine größere Streuung der Ausgangsentwürfe im Entwurfsraum erreicht.

7.6 Optimierung auf einem Kriging Modell

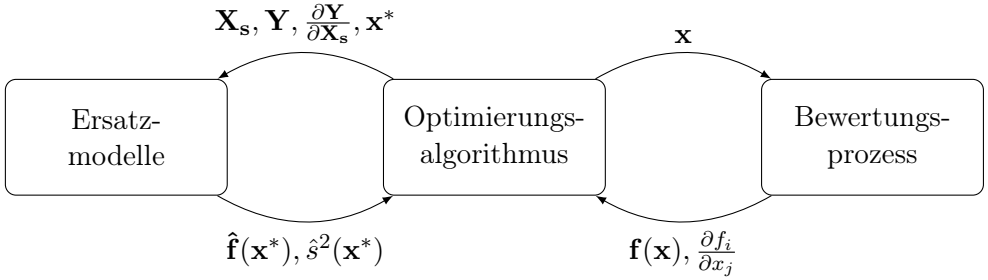


Abbildung 7.3: Optimierung mit einem gradientenbasierten Ersatzmodell

Nachdem ein initiales Ersatzmodell bereitsteht, kann dieses genutzt werden, um Vorhersagen $\widehat{f}(x^*)$ für verbesserte Entwürfe zu produzieren, welche dann durch den Bewertungsprozess tatsächliche Größen für die Bewertungsfunktionen $f(x^*)$ zugewiesen bekommen. Hierfür wird auf dem Ersatzmodell optimiert, also ein Optimierungsverfahren gestartet, welches als Funktionswerte nur die Vorhersagen des Ersatzmodells verwendet. Da die Vorhersage eines Wertes mit dem Ersatzmodell eine vergleichsweise günstige Operation darstellt, ist die Effizienz dieses Optimierungsalgorithmus zweitrangig. Hier werden evolutionäre Algorithmen verwendet.

Da das Ersatzmodell nur eine Approximation des Simulationsmodells darstellt, wird es zwischen $\widehat{y}$ und y eine Diskrepanz geben. Diese kann vermindert werden, indem die Datenbasis um die neu gewonnenen Bewertungen erweitert wird

$$Y^+ = [f(x_1), ..., f(x_D), f(x^*)]$$

und anschließend ein verbessertes Ersatzmodell erzeugt wird. Dabei gilt es zwei gegensätzliche Aufgaben zu erfüllen: Zum einen sollen erwartete lokale Minima möglichst exakt getroffen werden. Man sagt auch, dass ein Minimum möglichst weit ausgeschöpft werden soll (engl. exploitation). Zum anderen soll der Vorhersagefehler in Bereichen mit wenig Datenpunkten verringert werden, um Minima fernab der Ausgangspunkte zu identifizieren (engl. exploration).

Ausschöpfung eines Minimums wird erreicht, indem neue Punkte dort hinzugefügt werden, wo $\widehat{y}$ minimal ist. Exploration geschieht durch Hinzufügen neuer Punkte, für welche die vorhergesagte Unsicherheit (Gl. 7.16) besonders groß ist. Konzentriert man sich nur auf Ausschöpfung, wird das Verfahren gegen ein lokales Minimum konvergieren. Fokussiert man lediglich auf Exploration, wird man in der Regel ein vorhandenes Mini-

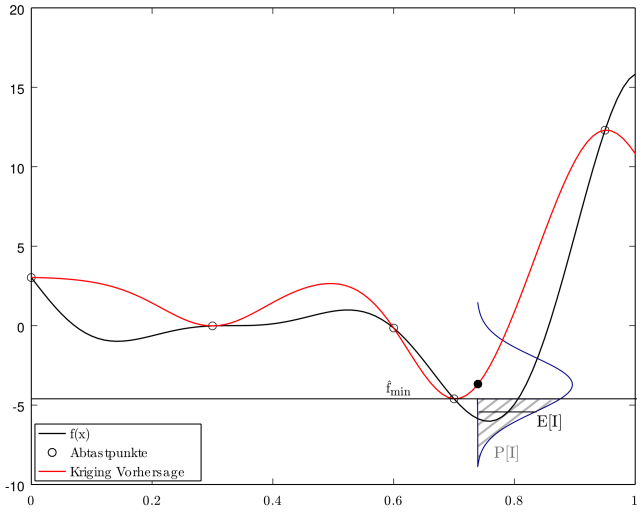


Abbildung 7.4: Illustration des Expected Improvement Kriteriums (nach [38]).

mum nicht ausschöpfen. Für einen Vergleich verschiedener Strategien zur Bestimmung auszuwertender Punkte siehe Jones [156].

Zur Balancierung dieser gegensätzlichen Ziele schlägt Jones [32] das Verfahren des „Expected Improvement“ vor: Der Funktionswert des bisher besten Punktes ist gegeben durch $\hat{f}_{\min} = \min(y_1, ..., y_M)$. Hat man einen weiteren Punkt x^* gefunden, für den das Ersatzmodell eine Verbesserung $\hat{f}(x^*) < \hat{f}_{\min}$ vorhersagt, so kann sich nach der Auswertung herausstellen, dass die Vorhersage des Ersatzmodells zu optimistisch war und x^* keine Verbesserung der Zielfunktion bringt.

Die Unsicherheit der Verbesserung wird als Zufallsvariable $I(x)$ modelliert, die von der gaußverteilten Zufallsvariable „Kriging Vorhersage“ $\hat{y}$ abhängt:

$$I = \max(f_{\min} - \hat{y}(x)), 0). \tag{7.28}$$

Die erwartete Verbesserung im Zielfunktionswert für diesen Punkt ist dann der Erwartungswert $E[I(x)]$. Im Falle einer Zielfunktion gilt (vgl. [38]):

$$E[I(x)] = (f_{\min} - \hat{y}(x)) \cdot \Phi\left(\frac{f_{\min} - \hat{y}(x)}{\hat{s}}\right) + \hat{s}\phi\left(\frac{f_{\min} - \hat{y}(x)}{\hat{s}}\right). \tag{7.29}$$

Wobei Φ und ϕ die Verteilungs- bzw. Wahrscheinlichkeitsdichtefunktion der Gauß-Verteilung bezeichnen. Sie kann als Schwerpunkt der Fläche interpretiert werden, welche vom Graph der Gaussverteilung unterhalb dem aktuell vohergesagten globalen Minimum $\hat{f}_{\min}$ eingeschlossen wird (s. Abb. 7.4).

Zur Auswahl eines neuen, zu bewertenden Punktes, wird der Punkt gesucht, welcher die erwartete Verbesserung maximiert. Dies kann auf zwei Arten geschehen. Durch einen

niedrigen vorhergesagten Funktionswert $\hat{y}$ oder eine hohe vorhergesagte Standardabweichung $\hat{s}$.

Punkte, die das Expected Improvement über einen niedriges $\hat{y}$ minimieren, sorgen für die Ausschöpfung vorhergesagter Minima. Punkte mit hoher Standardabweichung der Vorhersage $\hat{s}$ sind für die Exploration unbekannter Gebiete der Funktion verantwortlich. Eine Verschiebung der Balance beider Ziele kann durch eine Obergrenze von $\hat{s}$ während der Optimierung auf dem Ersatzmodell erreicht werden.

7.7 Kriging Modelle in Mehrzieloptimierungen

In Abschnitt 2.3 wurde auf die speziellen Probleme von Mehrzieloptimierungen eingegangen. Während das Expected Improvement für Einziel–Optimierungen analytisch berechnet werden kann, ist die Situation für Mehrzieloptimierungen komplexer. Vorgehensweisen zur Erweiterung auf die Pareto-Optimierung mit mehreren Zielen stellt Keane vor [164]. Einen Vergleich verschiedener neuerer Verfahren liefern Wagner et al. [165].

In dieser Arbeit wird der bereits vorgestellte Volumen-Gewinn aus dem statistisch vorhergesagten Member gebildet. Dieser Volumenzugewinn kann, analog zum eindimensionalen Fall, als Zufallsvariable ausgedrückt werden. Ihr Erwartungswert wird dann als erwarteter Volumenzugewinn (engl. Expected Volume Gain) bezeichnet.

Häufig möchte man parallel mehrere der aufwändigen Auswertungen gleichzeitig stattfinden lassen. Soll nun einer, oder mehrere, neue Bewertungspunkte gesucht werden, müssen diese zusammen mit allen laufenden Bewertungen das Expected Volume Gain minimieren. Man sucht also Vorschläge so dass diese zusammen mit den bereits laufenden Bewertungen das Expected Volume Gain gemeinsam maximieren.

Der gemeinsame Erwartungswert des Volumenzugewinns aller ausstehenden Bewertungen, wird mittels Monte-Carlo Simulation durch Ziehung aus der multivariaten Normalverteilung berechnet. Details hierzu finden sich bei Aulich et al. [34].

7.8 Gründe für die Wahl des Kriging-Ansatzes

Der hier maßgebliche Vorteil von Ersatzmodellen ist, dass sie unanfällig gegen das gelegentliche Fehlschlagen der Auswertungsprozesskette sind. Einige Optimierungsverfahren erfordern die Auswertung von Funktionswerten und Gradienten an vorgegebenen Punkten des Entwurfsraums. In praxisrelevanten Bewertungsprozessen können jedoch nicht an allen Punkten des Entwurfsraums Zielfunktionswerte oder Gradienten berechnet werden. Das liegt daran, dass einzelne Schritte der Auswertungskette fehlschlagen können. Beispielsweise falls ein Netz nicht generiert werden kann, oder ein iteratives Lösungsverfahren nicht konvergiert. Man benötigt Verfahren, die durch solche Definitionslücken nicht aufgehalten werden.

Ein Ersatzmodell ist nicht auf Funktionswerte an bestimmten Positionen gebunden, es werden vielmehr die erfolgreich durchgeführten Beobachtungen zu einem Modell vereinigt. Unter der Annahme, dass Bereiche für die der Bewertungsprozess fehlschlägt auch

keine optimalen Stellen der Zielfunktion sind, können Optimierungen somit unbeeinträchtigt einzelner Fehler der Auswertungsprozesse erfolgreich durchgeführt werden. Im Falle gradientenbasierter Optimierungen auf Basis des adjungierten Ansatzes kann es auch dazu kommen, dass einzelne partielle Ableitungen nicht berechnet werden können. Im GEK können in diesem Fall einfach die entsprechenden Einträge in Y_s, $\widetilde{\mathbf{\Psi}}$ und $\widetilde{c}$ weggelassen werden. Dennoch wird ein zu den verbleibenden Informationen passendes Ersatzmodell produziert.

Ein weiteres wichtiges Entscheidungskriterium ist die Tatsache, dass in praktischen Auslegungsprojekten nicht zu jeder Nebenbedingung effizient berechenbare Gradienten vorliegen. Dies ist regelmäßig der Fall, wenn Nebenbedingungen mit Programmen von Dritt-Herstellern berechnet werden, welche keinen adjungierten Löser mitbringen und zur algorithmischen Differenzierung nicht zugänglich sind, da sie nicht im Quellcode vorliegen.

Der Grund für die Wahl von Kriging als Ersatzmodellierungstechnik ist die Fähigkeit unterschiedlich aktive Entwurfsvariablen zu modellieren, wie sie für technische Optimierungen typisch sind.

Der statistische Ansatz des Krigings erlaubt die native Formulierung des Expected Improvements welches das bei Funktionen mit mehreren Minima die Balance zwischen Exploration und Exploitation erlaubt.

Die Eigenschaft, dass Trainingspunkte interpoliert werden, ist oft wünschenswert. Sind die Ausgabegrößen der Simulation jedoch mit Rauschen überlagert, kann durch Regularisierung der Kovarianzmatrix das Kriging leicht in ein Regressionsmodell umgeformt werden, welches für Optimierungen auf verrauschter Funktionen besser geeignet ist (vgl. [166]). Als Quellen für Rauschen in numerischen Simulationen gilt insbesondere unvollständige Konvergenz iterativer Verfahren.

Der wesentliche Nachteil des Krigings besteht in den hohen Kosten beim Training, also der Optimierung der Maximum-Likelihood Funktion. In jeder Iteration muss dort die Korrelationsmatrix faktorisiert werden, welche insbesondere beim Gradient Enhanced Kriging sehr groß und häufig schlecht konditioniert werden kann.

Die Entscheidung für den Ansatz des Kriging-Verfahrens entsteht daher aus der Abwägung des Trainingsaufwands und der Kosten einzelner Zielfunktionsauswertungen. In der Literatur finden sich vereinzelte Einschätzungen bis zu welcher Problemgröße Kriging geeignet ist (bspw. $M < 1000$ in [30], bzw. $N < 20$, $D < 500$ Tabelle 3.1 in [38]). Diese Einschätzungen werden allerdings bereits von den Autoren als Orientierungshilfe bezeichnet und müssen im Kontext der verfügbaren Rechenleistung zum Zeitpunkt der Aussage gesehen werden. Weiterhin muss der Einsatzzweck berücksichtigt werden, insbesondere ob eine global gute Vorhersage benötigt wird, oder nur das möglichst schnelle Auffinden von Minima. Außerdem ist eine der Stärken des Kriging-Ansatzes, dass er aktive Variablen gut von weniger aktiven Variablen unterscheiden kann und somit den Trainigs und Optimierungsaufwand auf einen Unterraum beschränken kann. Für die Optimierungsprobleme in der Turbomaschinenauslegung hat sich Kriging im praktischen Einsatz als geeigneter Ansatz auch bei deutlich größeren Matrixdimensionen herausgestellt [135].

8 Optimierungsstudie am gegenläufigen Fan

Die in den vorherigen Abschnitten erarbeitete Methode zur gradientenbasierten Optimierung wird in diesem Kapitel auf ein Beispiel aus der Auslegungspraxis angewandt, um die Umsetzbarkeit zu demonstrieren. Hierzu wird das CRISPmulti Projekt [74] herangezogen, welches zum Ziel hatte, das Potential gegenläufiger Gebläse-Stufen (Fan) gegenüber konventionellen Entwürfen zu bewerten. Zu diesem Zweck wurde für jedes der beiden Konzepte eine Auslegungsoptimierung durchgeführt und die Ergebnisse experimentell untersucht. Die Optimierung diente hier also dazu, zwei Konzepte bei optimaler Ausführung vergleichen zu können.

8.1 Gegenläufige Fans

Eine der wichtigsten Herausforderungen für die Luftfahrt besteht in der Verringerung des Treibstoffverbrauchs und der Emmissionen. Die Europäische Kommission hat für den Luftverkehr unter dem Titel Flightpath 2050 [167], unter anderem, folgende Ziele formuliert: Eine Reduktion des CO_2 um 75%, der NO_x-Emissionen um 90% pro Passagierkilometer sowie des wahrgenommenen Lärms um 65% im Vergleich zu neuen Flugzeugen des Jahres 2000. Dies wurde durch den Luftfahrtforschungsbeirat ACARE (Advisory Council for Aviation Research and Innovation in Europe) in die Forschungsagenda SRIA (Strategic Research and Innovation Agenda) übersetzt, welche einen erheblichen Anteil dieser Reduktionen durch Verbesserungen des Triebwerks vorsieht, unter anderem eine Reduktion des Treibstoffverbrauchs und der CO_2-Emissionen um 40%.

Neuartige Konzepte könnten zu den geforderten Reduktionen beitragen. Eine Möglichkeit zur Effizienzsteigerung des Triebwerks besteht in der Verwendung gegenläufiger Fan-Stufen (Counter Rotating Fan CRF) [168]. Gegenläufige Fan-Stufen vereinen Eigenschaften eines klassischen Turbofan-Triebwerks mit denen eines gegenläufigen offenen Rotors (Counter Rotating Open Rotor - CROR). Wie der CROR besitzen CRF zwei direkt hintereinander angeordnete, entgegengesetzt rotierende Schaufelreihen. Im Unterschied zum CROR ist der gegenläufige Fan jedoch von einem Gehäuse umgeben. Dieses bringt zwar zusätzliche Verluste durch vergrößerte Wandfläche und einen erhöhten Widerstandsbeiwert mit sich, senkt jedoch die Lärmemissionen, welche insbesondere durch die Interaktion der Rotorströmungen mit hohen Relativgeschwindigkeiten verursacht werden. Darüber hinaus erlaubt das Gehäuse die akustisch wünschenswerte Positionierung des Triebwerks an der Flügelunterseite.

Im Vergleich zum klassischen Fan besteht der Vorteil in der Ersetzung des passiven Leitrades durch eine zweite Rotor-Reihe, so dass der Totaldruck-Aufbau der Fan-Stufe auf

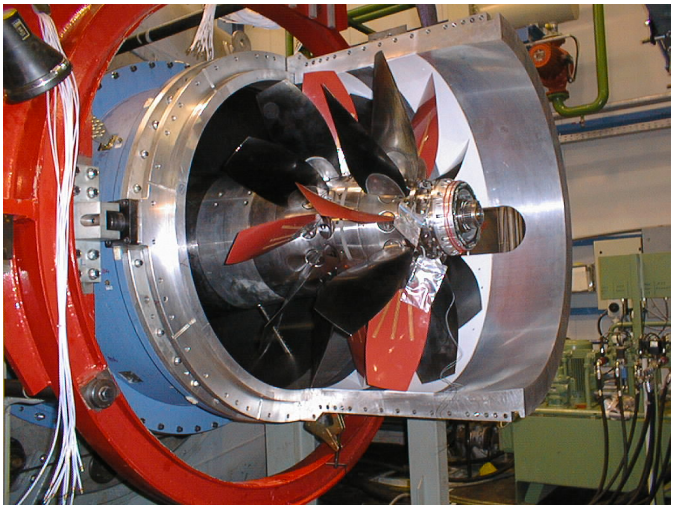


Abbildung 8.1: Instrumentiertes 1m-Prüfstandsmodell des gegenläufigen Fans CRISP auf dem Verdichterprüfstand M2VP des DLR.

zwei Reihen verteilt werden kann, wodurch die aerodynamische Belastung der einzelnen Schaufelreihen geringer ausfallen kann. Dies erlaubt bei gleichbleibendem Durchmesser und Axialgeschwindigkeit eine geringere Zahl von Schaufeln pro Rotor zu verwenden, wodurch das Flächenverhältnis verbessert wird [169]. Zum Antrieb der gegenläufigen Fan-Stufe kommen entweder zwei entgegengesetzt rotierende Niederdruckturbinenstufen oder ein Getriebe zum Einsatz. Die Markteinführung des Geared Turbofans (GTF) für den A320neo zeigt die technische Reife der hierfür nötigen Getriebe.

Die MTU Aero-Engines AG vermutete 2007 eine mögliche Senkung des Treibstoffverbrauchs um bis zu 20% im Vergleich zu Triebwerken des Jahrs 2000 durch eine Kombination von Effizienzsteigerungsmaßnahmen, welche auch gegenläufige Fans enthalten (vgl. Geschäftsbericht der MTU AG 2007 [170]).

8.2 Das CRISPmulti Auslegungsprojekt

Zur Bewertung des Potentials gegenläufiger Fans unter modernen Auslegungsverfahren wurde für eine gegenläufige und für eine konventionelle Fan-Stufe, unter gleichen Anforderungen, eine Auslegungsoptimierung durchgeführt und der jeweils beste Entwurf gebaut und experimentell untersucht. Diese Arbeiten werden in der Dissertation von Lengyel-Kampann beschrieben [74]. Die optimierungsbasierte Auslegung dient darin als Forschungswerkzeug um die technischen Potentiale eines neuartigen Konzepts zu bewerten. Die Untersuchung von Lengyel-Kampmann stellt eine mögliche Verbesserung des Wirkungsgrads der Fan-Stufe durch Gegenläufigkeit von bis zu 4% in Aussicht.

Im Weiteren wird die Auslegung der gegenläufigen Stufe aus [74] vorgestellt. Die Designoptimierung basiert auf einer gegenläufigen Fan-Stufe CRISP (Counter Rotating Integrated Shrouded Propfan), welche Anfang der 1990er Jahre ausgelegt und als Modell

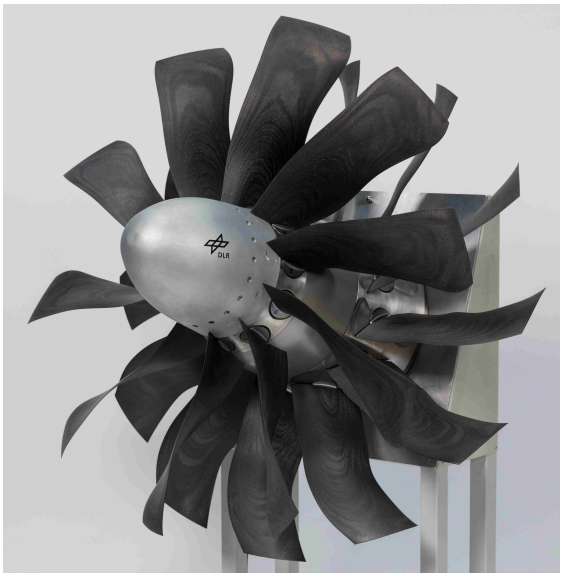


Abbildung 8.2: Prüfstandsmodell CRISPmulti Rotors.

mit einem Meter Durchmesser auf dem Verdichterprüfstand M2VP des Instituts für Antriebstechnik am DLR vermessen wurde [171] (s. Abb. 8.1).

Die Neuauslegung CRISPmulti berücksichtigt insbesondere die Fortschritte in Materialien und Bauweisen durch kohlefaserverstärkte Kunststoffe [172] sowie die Möglichkeit, nun dreidimensionale Strömungssimulationen als Auslegungswerkzeug zu verwenden. Das Prüfstandsmodell, welches auf Basis dieser Optimierung gefertigt wurde, ist in Abbildung 8.2 dargestellt.

Ein Auslegungsprojekt für ein Prüfstandsexperiment muss eine Vielzahl interdisziplinärer Anforderungen berücksichtigen. Für CRISPmulti waren dies [74]:

- Hoher isentroper Wirkungsgrad im aerodynamischen Auslegungspunkt (ADP)
- Geringer Restdrall am Austritt
- Einhaltung statischer Festigkeitskriterien: Minimale Profildicke, keine Minima im Dickenverlauf, Einhaltung einer maximalen Vergleichsspannung nach von Mises.
- Einhaltung dynamischer Festigkeitskriterien bezüglich Eigenfrequenzen
- Getriebebedingtes, maximal zulässiges Drehmomentverhältnis der Rotoren

Der Designraum für die beschriebenen Auslegungen ist mit 80-100 freien Variablen typisch für die Auslegung einstufiger Turbomaschinenkonfigurationen.

Für Optimierungsverfahren ist eine solche Parameterzahl jedoch herausfordernd. In der gradientenfreien, ersatzmodellbasierten Optimierung wurden mehr als 2000 Designs erfolgreich bewertet. Diese Zahl ist im Verhältnis zur Größe des Designraums klein, im

Hinblick auf die Rechenkosten mit einer Stunde auf einem Clusterknoten pro Bewertung jedoch bereits erheblich. Die Diskrepanz zwischen Entwurfsraumgröße und Zahl zulässiger Zielfunktionsauswertungen konnte durch effiziente Ersatzmodellierungstechniken überbrückt werden. Eine weitere Ressourcenreduktion für diese Klasse von Optimierungsproblemen ist für zukünftige Auslegungen dennoch wünschenswert. Eine Auslegung könnte dann in kürzerer Zeit, mit mehr Parametern oder höherem Modellierungsgrad, durchgeführt werden.

Abseits von der hier beschriebenen Anwendung auf eine Fan-Stufe sind die oben beschriebenen Techniken auch auf Verdichter und Turbinen übertragbar.

8.2.1 Oberflächengenerierung

Für die Optimierung von Turbomaschinenkomponenten müssen die umströmten Oberflächen systematisch verändert werden können. Hierbei kommt es darauf an, alle gewünschten Oberflächen als Funktion reeller Designvariablen zu beschreiben. Die Abbildung eines endlich-dimensionalen Vektors auf dreidimensionale Flächen wird Parametrisierung genannt. In der Literatur werden verschiedene Parametrisierungen vorgeschlagen [173], die sich in folgende Klassen einteilen lassen:

1. CAD basierte Beschreibung der Schaufeln durch Familien von parametrisierten Konstruktionsprofilen und deren räumliche Anordnung (Fädelung)

2. Beschreibung durch Modifikationen von Referenzprofilen und deren räumliche Anordnung (bspw. Hicks-Henne Deformationen)

3. Freiformdeformation einer vernetzten Referenzschaufel mittels räumlicher Splines [174]

4. Variation der Rechennetzknoten auf der Oberfläche einer Referenzgeometrie

Die Verfahren sind in der obigen Aufzählung aufsteigend nach Parameterzahl sortiert: Verfahren (1) benötigt typischerweise eine Größenordnung weniger Parameter zur Beschreibung einer Schaufel als Vorgehensweise (4). Gleichzeitig sind sie auch absteigend nach Problemspezifizität sortiert: Für die ersten beiden Verfahren werden Entwurfswerkzeuge benötigt, welche auf aerodynamische Profile und Formen spezialisiert sind. Insbesondere bei Vorgehen nach Methode (1) wird bereits Wissen über zulässige Formen und deren effiziente Beschreibung in das Entwurfswerkzeug integriert. Dagegen sind die Verfahren (3) und (4) nicht spezifisch für den Turbomaschinenentwurf, sondern können leicht auf vollkommen andere Entwurfsprobleme angewendet werden, da sie nur die Konzepte eines Rechennetzes und darin befindlicher fester Oberflächen besitzen müssen.

Zur Durchführung der Optimierungsstudie in dieser Arbeit wird ausschließlich Vorgehensweise (1) angewandt.

Hierfür ist eine Reihe von Gründen entscheidend. Zunächst ist wichtig, dass nur diese Form der Parametrisierung eine direkte Weiterverarbeitung des optimierten Designs in der für die Auslegungspraxis relevanten Darstellung ermöglicht. Das Ergebnis der unspezifischen Parametrisierungsmethoden (3) und (4) muss für eine solche Weiterverarbeitung

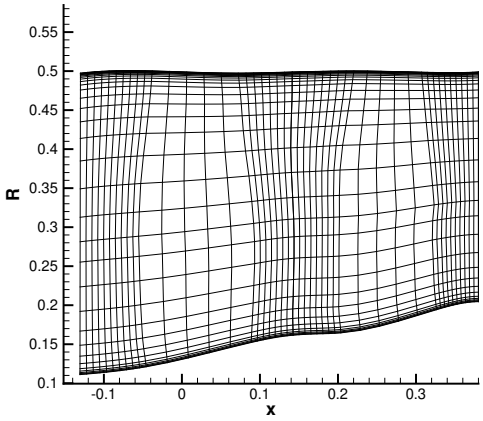


Abbildung 8.3: Konstruktions-Netz der S2-Ebene

in die problemspezifische Parametrisierung konvertiert werden. Dies ist rechenaufwändig, und mit einem möglichen Verlust der Optimalität verbunden.

Klassische Entwurfsparameter wie Winkel, Längen und Materialdicken erlauben zudem eine bessere Interpretation des Optimierungsergebnisses, da sie direkt mit der profilbasierten Auslegungstheorie korrespondiert.

Für die Verwendung der diskretisierten Oberflächenpunkte wie in (4) spricht zwar der größere Raum beschreibbarer Entwürfe und die verhältnismäßig einfache Umsetzbarkeit durch Werkzeuge zur glatten Netzdeformation, jedoch existieren in einem solchen Designraum viele Entwürfe, welche aufgrund einfacher aerodynamischer oder mechanischer Überlegungen offensichtlich ungeeignet sind. Punktweise Parametrisierungen wie (4) benötigen in der Regel eine Glättung des Designraums wie beispielsweise in [3, 175].

Zudem erfordern globale Änderungen eines aerodynamischen Profils, beispielsweise die Umstaffelung, die passende simultane Änderung aller Punkte dieses Profils. Diese allein über lokale Parameter abzubilden ist ein hochdimensionales Einstellungsproblem, während es in einer problemangepassten Parametrisierung typischerweise durch einen einzelnen Parameter dargestellt werden kann. So kommen Überblicksartikel zu diesem Thema ebenfalls zu der Schlussfolgerung, dass eine problemspezifische Parametrisierung vorzuziehen ist, wenn sie verfügbar ist [176].

8.2.2 Oberflächendarstellung

Die Oberflächengenerierung welche in dieser Arbeit Anwendung findet, orientiert sich an verbreiteten Vorgehensweisen für den Entwurf von Turbomaschinenbeschaufelungen. Da eine große Zahl technischer Variationen der Vorgehensweise existieren, wird hier die verwendete Vorgehensweise dargestellt. Zunächst wird der rotationssymmetrische Ringraum mittels Nabe- und Gehäusekurven in Zylinderkoordinaten (x, R, θ) dargestellt. Die Beschreibung findet in einer (x, R)-Ebene statt (S2-Ebene nach Wu [177]). Naben und

Gehäuseform werden durch zwei Kurven γ_{Hub} und γ_{Casing} in dieser Ebene definiert. Zwischen ihnen wird eine Kurvenschar definiert, welche als Approximation der zu erwartenden Stromlinien verlaufen (Konstruktionsstromlinien).

Diese Kurven werden so diskretisiert, dass ein strukturiertes Netz entsteht. Die Konstruktionsstromlinien definieren Rotationsflächen (S1-Flächen nach [177]). Diese können in das winkelerhaltende Koordinatensystem (m', θ) transformiert werden, wobei m' einen Lauflängenparameter der jeweils definierenden Kurve darstellt.

Auf mehreren der Rotationsflächen werden Konstruktionsprofile als C^2-stetige, geschlossene Kurve durch jeweils vier B-Spline Kurven [178] definiert, welche den aerodynamischen Funktionsbereichen des Profils entsprechen: Vorderkante, Hinterkante, Saugseite und Druckseite. Die Beschreibung der Profilkontur besteht zunächst aus aerodynamisch motivierten Profilparametern. Da die B-Splines mit beliebiger Anzahl Freiheitsgrade definierbar sind, können noch weitere Freiheitsgrade übrig bleiben, welche insbesondere für die Definition des Splines der Saugseite bedeutsam sind.

Die Definition der Druckseite wird abhängig von der Saugseite durch eine Profildickenverteilung generiert, welche durch einen Spline definiert wird. Der Vorteil gegenüber einer freien Vorgabe der Druckseite ist ein Designraum mit einem höheren Anteil aerodynamisch sinnvoller Geometrien und der Möglichkeit, festigkeitsmechanisch bedeutsame Parameter wie Füllungsgrade, maximale Dicke und Dickensteigerungen direkt im Design vorzugeben.

Nachdem die einzelnen Profile auf den Konstruktionsstromflächen definiert wurden, werden diese in der (m', θ)-Ebene gedreht (gestaffelt) und in eine gewünschte axiale Länge gebracht. Diese zweidimensionalen Profile werden entlang einer räumlichen Kurve übereinander angeordnet (gefädelt). Die Oberfläche der Schaufel wird als ebener Spline dargestellt, indem die Profilsplines mittels einer zusätlichen Spline-Funktion in Spannweitenrichtung verbunden werden.

Zur glatteren Wiedergabe der Schaufeloberfläche ist es sinnvoll hierbei eine gewisse Mindestzahl an Profilsplines zu verwenden. Jedoch möchte man nicht jedes Profil einzeln parametrisieren, da sich eine zu hohe Parameterzahl ergeben würde. Aus diesem Grund werden zwischen den frei parametrisierbaren Profilen weitere abhängige Profile durch Interpolation der Parameter erzeugt.

Die Konfiguration wird durch 81 freie Parameter beschrieben, welche auf 3 parametrisierte Profile je Rotor verteilt sind. In Tabelle 8.1 sind die Parameter zusammengefasst.

8.2.3 Netzgenerierung

Ausgehend von den Oberflächen der Schaufeln wird ein Rechennetz für die Strömungssimulation generiert. Es kommen blockstrukturierte Rechennetze zum Einsatz, da sie aus Gründen der Genauigkeit und Recheneffizienz vorteilhaft sind. Die Netze werden mittels G3DHexa [179] definiert und erzeugt.

Zur Netz-Erzeugung wird das 3D-Modell der Schaufel an einer vorgegebenen Position im Strömungskanal angeordnet und mit dessen Konturen verschnitten. Hierbei werden auch Gehäuse- bzw. Nabenspalte definiert.

	Anzahl	Beschreibung
Ringraum	3	Nabenkontrollpunkte Verschiebung in axialer Richtung
	5	Nabenkontrollpunkte Verschiebung in radialer Richtung
	5	Gehäusekontrollpunkte Verschiebung in radialer Richtung
Rotoren	2	Radiale Verschiebung des 2. Konstruktionsprofils
	6	Staffelungswinkel
	12	Winkel an Profilvorderkanten
	12	Winkel an der Profilhinterkanten
	12	Halbachsen Profilvorderkanten
	6	Radien an Hinterkanten
	6	Profilsehnenlängen
	12	Spline-Kontrollpunkte auf Saugseiten.

Tabelle 8.1: Zusammenfassung der verwendeten Parameter

Daraufhin werden die ebenen Netze auf den S1-Flächen von Nabe und Gehäuse erzeugt. In dieser Arbeit wird eine OCH-Topologie verwendet . Das Schaufelnetz-Volumen entsteht durch die Interpolation zwischen diesen gegebenen Netzen.

Für die Optimierung wird eine Passage beider Rotoren mit insgesamt ca. 688 000 Zellen vernetzt. Die Modellierung der Wandgrenzschicht erfolgt durch Wandfunktionen, der dimensionslose erste Wandabstand wurde daher mit $y^+ \approx 30$ entsprechend gewählt. Der Übergang der Schaufelblätter in die Nabe wurde senkrecht, das heißt ohne Ausrundungsradien dargestellt. Der Vernetzungsprozess ist automatisiert, so dass nach erstmaligen Aufsetzen, topologisch gleiche Netze für eine große Bandbreite von Variationen des Ursprungsentwurfs ohne manuellen Eingriff erzeugt werden können.

Der Vernetzungsprozess und die Definition des initialen Netzes entstammt im wesentlichen Teilen der Arbeit von Lengyel-Kampmann [74] und ist typisch für den Modellierungsgrad innerhalb einer Optimierung. Die Verifikation mittels feinerer Netze wurde ebendort durchgeführt und wird daher hier ausgelassen.

8.2.4 Strömungssimulation

Für die aerodynamische Bewertung werden die dreidimensionalen stationären Navier-Stokes Gleichungen entsprechend Abschnitt 3.2 verwendet. Diese sind im stationären Löser des Turbomaschinen-Simulationspakets TRACE implementiert. Eine Validierung stationärer Simulationen für gegenläufige Fans wurde per Vergleich mit instationären Simulationen und Experimenten von Lengyel-Kampann [180] für eine vergleichbare Konfiguration (CRF) vorgenommen.

Als Turbulenzmodell kommt hier das $k - \omega$ Zwei-Gleichungs Turbulenzmodell nach Wilcox [97] zum Einsatz. Es wird durch Staupunktkorrektur nach Kato-Launder und Rotationstermen nach Bardina sowie Korrekturfaktoren für Kompressibilitätseffekte ergänzt. Die Grenzschichten werden als vollturbulent angenommen. Die Wandgrenzschicht wird nicht aufgelöst, sondern mittels Wandfunktionen dargestellt.

Am Eintritt wird eine Zuströmung mit Mach 0.6 einem Totaldruck von 101 325.0 Pa

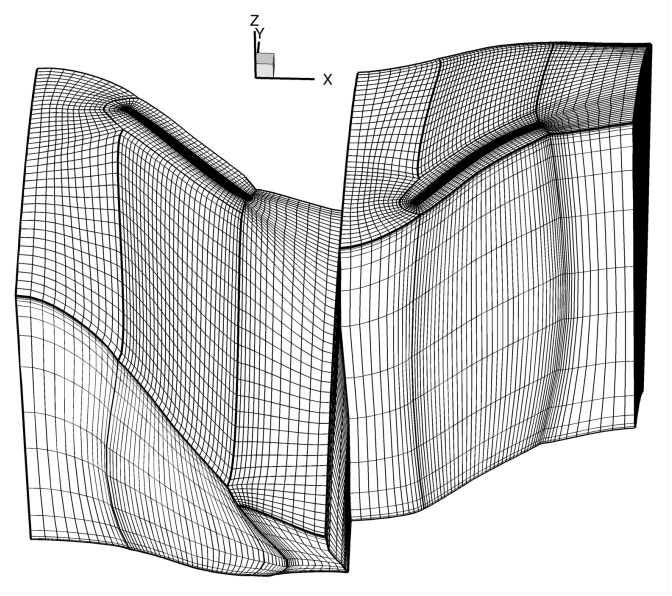


Abbildung 8.4: Volumen-Netz zur Strömungssimulation (nur jede zweite Linie dargestellt)

sowie einer Totaltemperatur von 288.0 K vorgeschrieben, die Reynoldszahl bezogen auf die Schaufelsehnenlänge beträgt dort 3×10^6.

Die Schaufelreihen sind durch eine Mischungsebene gekoppelt. Am Austritt wird ein statischer Gegendruck entsprechend des gewählten Betriebspunkts vorgegeben.

Die CFL Zahl im Auslegungspunkt beträgt 30, im pumpgrenznahen Punkt 15.

8.2.5 Zielfunktionen und Nebenbedingungen

Der Ausgangsentwurf für diese Optimierung besitzt eine Zuströmgeschwindigkeit von Mach 0.6. Die Strömung ist im Blattspitzenbereich transsonisch mit einer Vorstoß-Machzahl von bis zu 1.2. Die Gesamtstufe produziert einen Massenstrom von 159 kg/s bei einem Totaldruckverhältnis von 1.29. Das Hauptziel der Auslegung ist eine Maximierung des isentropen Wirkungsgrades

$$\eta_{\mathrm{is}} = \frac{\Delta h_{\mathrm{t,is}}}{\Delta h_{\mathrm{t}}} = \frac{h_{\mathrm{t,abs,out,is}} - h_{\mathrm{t,abs,in}}}{h_{\mathrm{t,abs,out}} - h_{\mathrm{t,abs,in}}} \tag{8.1}$$

mit

$$h_{\mathrm{t,abs}} = e + \frac{p}{\rho} + \frac{1}{2} v_{\mathrm{abs}}^2 \tag{8.2}$$

im aerodynamischen Auslegungspunkt (ADP). Wobei das Subskript t Totalgrößen kennzeichnet, abs Größen im Absolutsystem, is isentrope Wert und in sowie out für Größen an Ein- respektive Austrittsrand stehen. Die spezifische innere Energie wird mit e bezeichnet, p ist der statische Druck und v_{abs} der Betrag der Geschwindigkeit im Absolutsystem.

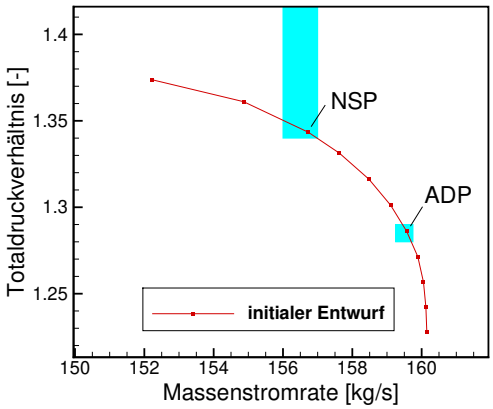


Abbildung 8.5: Arbeitslinie des initialen Entwurfs mit Nebenbedingungen (blaue Felder)

Der ADP ist durch einen statischen Druck am Austritt der Stufe definiert. Um den Betriebspunkt während der Optimierung nicht zu verändern, wird der Massenstrom durch eine Nebenbedingung auf einen Bereich von 0.5 kg/s festgehalten. Das Totaldruckverhältnis im Absolutsystem

$$\pi_{\mathrm{t,abs}} = \frac{p_{\mathrm{t,out,abs}}}{p_{\mathrm{t,in,abs}}} \tag{8.3}$$

mit

$$p_{\mathrm{t,abs}} = p\left(1 + \frac{\gamma - 1}{2} M^2_{\mathrm{abs}}\right)^{\frac{\gamma}{\gamma - 1}} \tag{8.4}$$

im ADP wird auf das Intervall 1.285 ± 0.005 festgelegt.

Um den Arbeitsbereich der Stufe durch die Optimierung des Wirkungsgrades nicht zu verkleinern, wird ein Pumpgrenzkriterium berücksichtigt. Dieses wird als zusätzliche Zielfunktion behandelt, um nach der Optimierung die Interaktion von Wirkungsgradgewinn und Arbeitsbereichseinschränkung quantifizieren zu können. Da die numerische Einschätzung des Arbeitsbereichs schwer durchzuführen ist, wird ein Pumpgrenzkriterium nach Cumpsty eingesetzt. Für dieses wird ein zweiter Betriebspunkt näher der Pumpgrenze (near stall point NSP) simuliert und die Steigerung des Totaldrucks zwischen ADP und NSP als Indikator für die Arbeitsbereichsweite herangezogen. In der hier vorgestellten Optimierung wird als zweite Zielfunktion gefordert, das Totaldruckverhältnis im NSP zu steigern. Der NSP wird über einen Gegendruck eingestellt, so dass dieser einen 2.5% niedrigeren Massenstrom aufweist als der ADP. Der NSP wird durch Nebenbedingungen im Massenstrom auf einen Bereich von 1 kg/s und im Totaldruckverhältnis auf eine Untergrenze von 1.34 beschränkt. Die zulässigen Betriebspunkte sind in der Arbeitslinie in Abb. 8.5 als blaue Flächen dargestellt. Eine Zusammenfassung der erläuterten Ziele und Nebenbedingungen befindet sich in Tab. 8.2.

Bezeichnung	Symbol	Ziel/Einschränkung
isentroper Wirkungsgrad im ADP	$\eta_{\text{is,ADP}}$	erhöhen/ $[0.75, \infty[$
Totaldruckverhältnis im NSP	$\pi_{\text{tot,NSP}}$	erhöhen/ $[1.34, \infty[$
Massenstrom im ADP	$\dot{m}_{\text{ADP}}$	$[159.25, 159.75]$ kg/s
Totaldruckverhältnis im ADP	$\pi_{\text{tot,ADP}}$	$[1.28, 1.29]$
Massenstrom im NSP	$\dot{m}_{\text{NSP}}$	$[156, 157]$ kg/s
maximale von Mises Spannung Rotor 1	$\sigma_{v,R1}$	$]-\infty, 400]$ MPa
maximale von Mises Spannung Rotor 2	$\sigma_{v,R2}$	$]-\infty, 400]$ MPa

Tabelle 8.2: Optimierungsziele und Nebenbedingungen

8.3 Sensitivitätenberechnung

8.3.1 Beschreibung des Berechnungsprozesses

Während die theoretischen Grundlagen der adjungierten Berechnung partieller Ableitungen in Kapitel 5 besprochen wurden, soll hier die praktische Implementierung dargestellt werden. Hierbei bestehen zwei wesentliche Herausforderungen. Erstens sind die Berechnungen über mehrere gekoppelte Programme verteilt. Daher muss zur Berechnung der partiellen Ableitungen auch die Kopplung der Programme adjungiert werden. Es müssen also nach der Adjungierung der einzelnen Programme auch Schnittstellen für die adjungierten Ergebnisse geschaffen werden.

Zweitens kann ein Teil der Prozesskette, nämlich die Abbildung von Designparametern in Rechennetze, normalerweise nicht adjungiert werden, da die Programme zur Geometriegenerierung und Netzerzeugung von Drittherstellern stammen.

Für die Netzgenerierung könnte man ersatzweise auf adjungierte Netzdeformation zurückgreifen [133]. Für die Geometriegenerierung gibt es Ansätze auch CAD-Bibliotheken algorithmisch zu differenzieren [181]. Jedoch bedingt dies die Ersetzung von Teilen einer etablierten Entwurfsprozesskette durch andere Programme, was in einem industriellen Umfeld nicht immer gewünscht oder möglich ist. Daher wird in dieser Arbeit die Abbildung der Entwurfsparameter auf Rechennetze als Black-Box behandelt und mittels finiter Differenzen abgeleitet.

Im folgenden wird die Ausführung und Kopplung der verschiedenen Prozessteile dargestellt. In den folgenden Darstellungen sind die Knoten der Graphen Dateien, welche von Programmen gelesen und geschrieben werden. Die Programme sind als beschriftete Pfeile repräsentiert.

Schematisch sieht die Abbildung von den Entwurfsparametern auf Rechennetze wie folgt aus (Abb. 8.6):

Die Entwurfsparameter α sind hierbei die in Tabelle 8.1 genannten Beschreibungen der umströmten Oberflächen. Aus diesen werden, wie in Abschnitt 8.2.2 beschrieben, zunächst NURBS-Oberflächen erzeugt und diese anschließend diskretisiert.

Durch den Netzgenerator wird das durchströmte Volumen mit einem strukturierten Rechennetz für die nachfolgende Strömungssimulation diskretisiert.

Der Simulationsprozess ist in Abbildung 8.7 dargestellt:

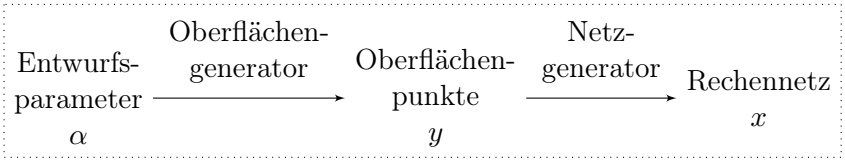


Abbildung 8.6: Prozesskette: Netzgenerierung

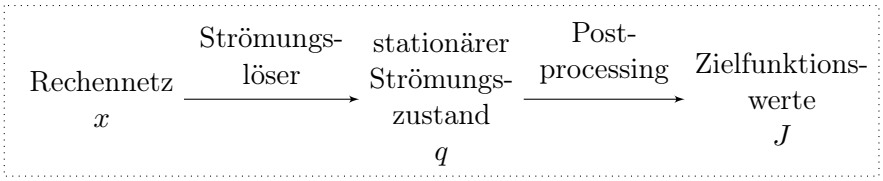


Abbildung 8.7: Prozesskette: primale Strömungssimulation

Die primale Strömungssimulation erzeugt aus dem Rechennetz x zunächst den dreidimensionalen Strömungszustand q. Dieser entspricht dem erweiterten Zustand welcher in Abschnitt 6.3.5 besprochen wurde. Aus dem Strömungsfeld werden im Schritt des Post-Processings integrale Bewertungsgrößen erzeugt. Randbedingungen, Initiallösung und Einstellungen des Lösungsverfahrens wurden in der obigen Darstellung der Übersichtlichkeit halber ausgelassen, da sie als konstant angenommen werden können.

Bis hierhin ist die Prozesskette unabhängig vom Adjungiertenverfahren und entspricht einem typischen Bewertungsprozess, wie er auch in der gradientenfreien Optimierung verwendet wird. In Abschnitt 8.4, wird ersichtlich, dass es auch in einer gradientengestützten Optimierung nötig ist, diesen Prozess getrennt von der Gradientenberechnung ausführen zu können.

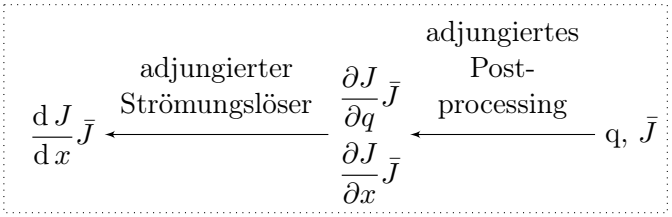


Abbildung 8.8: Prozesskette: adjungierte Strömungssimulation

Der Prozess zur Gradientenberechnung besteht aus zwei Teilen. Der erste Teil berechnet adjungiert die Ableitungen des Zielfunktionals nach allen Netzkoordinaten: Die Netzsensitivitäten. Der zweite Teil bildet diese dann mittels finiter Differenzen auf Sensitivitäten der Entwurfsparameter ab. Die Prozesskette zur adjungierten Strömungssimulation (Abb. 8.8) beginnt mit der Wahl des zu differenzierenden Funktionals durch die Wahl eines Auswertefunktionals $\bar{J}$, welche typischerweise einem Einheitsvektor entspricht, um nach genau einer Ausgabegröße abzuleiten.

Das adjungierte Post-Processing wird wie in Abschnitt 6.3.4 beschrieben, durchgeführt. Hierzu müssen die Operationen des primalen Post-Processings aufgezeichnet werden. Die Belegung der adjungierten Werte an den Ausgabegrößen bezeichnen wir mit dem Vektor $\bar{J}$, welcher typischerweise dem i-ten Einheitsvektor und damit der Wahl des i-ten Ausgabefunktionals entspricht. Durch Ausführung des adjungierten Post-Processings für diese Werte erhält man die benötigten Ableitungen. Man erhält den Gradienten des gewählten Zielfunktionals nach den Zustandsgrößen $\partial J_i/\partial q$ und $\partial J_i/\partial x$. Diese wiederrum sind, zusammen mit dem Zustand q, welcher als sogenannter Checkpoint vorliegt, Eingabedaten für den adjungierten Strömungslöser. Dieser berechnet mittels der in Abschnitt 6.3.5 beschriebenen Fixpunkt-Rückwärts-Iteration die Ableitungen des Funktionals nach allen Netzkoordinaten $\mathrm{d}J_i/\mathrm{d}x$.

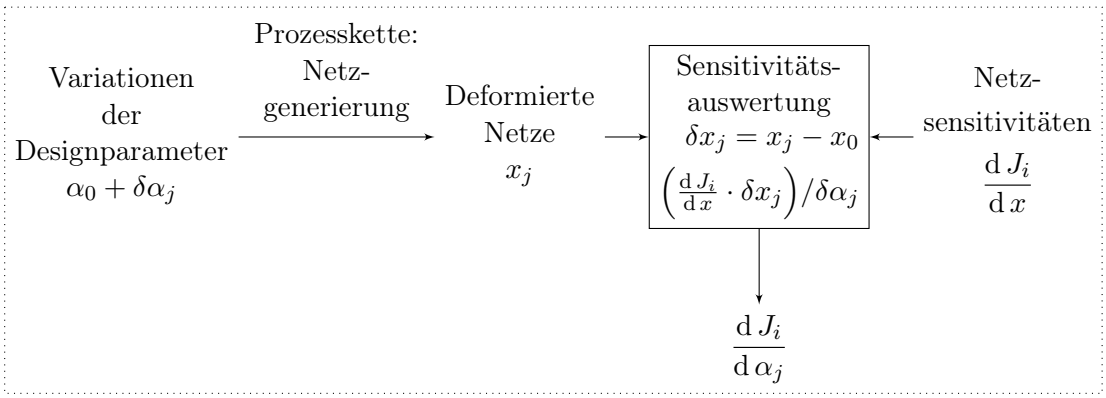


Abbildung 8.9: Prozesskette: Auswertung der Ableitungen nach Designparametern

Wir haben nun die Ableitungen nach allen Netzkoordinaten, und wollen diese auf Ableitungen der Entwurfsparameter abbilden. Dies geschieht durch Skalarprodukte der Netzsensitivitäten mit finiten Differenzen des Netzgenerierungsprozesses. Hierzu wird nacheinander jeder Entwurfsparameter variiert und daraus eine Netzvariation erzeugt, indem die Prozesskette zur Netzgenerierung durchlaufen wird. Man erhält für jeden Entwurfsparameter ein variiertes Netz, welcher eine Approximation von $\partial x/\partial\alpha$ darstellt, welche wir im Folgenden mit δx_j bezeichnen. Der Zeit- und Speicheraufwand dieses Prozessteils hängt natürlich linear von der Zahl der Parameter ab und kann bei großen Designräumen problematisch werden. Um die Gesamtzeit zu reduzieren, kann die Erzeugung deformierter Netze parallel zur Berechnung der Netzsensitivitäten durchgeführt werden. Zur Reduktion des Speicherbedarfs für die deformierten Netze ist es möglich, nur diejenigen Netzblöcke neu zu erzeugen und abzuspeichern, deren Netzknoten verschoben wurden. Dies ist insbesondere bei Rechennetzen mehrstufiger Turbomaschinen interessant, bei denen die Optimierungsparameter nur auf einen Teil der Stufen wirken. Für nicht vorhandene Blöcke im deformierten Rechennetz wird im folgenden Prozess angenommen, dass diese unverändert bleiben.

Die Auswertung besteht darin, dass zunächst die Verschiebung jedes Netzknotens δx_j aus der Differenz der Netzknoten des deformierten Netzes x_j und des Nominalnetzes x_0 berechnet wird. Dass sich auf die hier beschriebene Weise $\delta\alpha_j$ auf δx_j abbilden lässt,

basiert auf der Annahme, dass sich die Verschiebung jedes Netzknotens im Bereich der Variationsgröße linear verhält. Dies kann im Vorhinein durch Parameterstudien verifiziert werden. Die finale Parametersensitivität wird daraufhin durch Bildung des Skalarprodukts aus Netzsensitivität und dem soeben berechneten Netzdeformationsvektor gebildet

$$\left(\frac{\mathrm{d}\, J_i}{\mathrm{d}\, x} \cdot \delta x_j\right) / \delta \alpha_j. \tag{8.5}$$

Dies sind die Ableitungen der gewählten Zielfunktionale nach den gewählten Entwurfsparametern.

8.3.2 Auswirkung der Deformationsschrittweite auf die berechneten Sensitivitäten

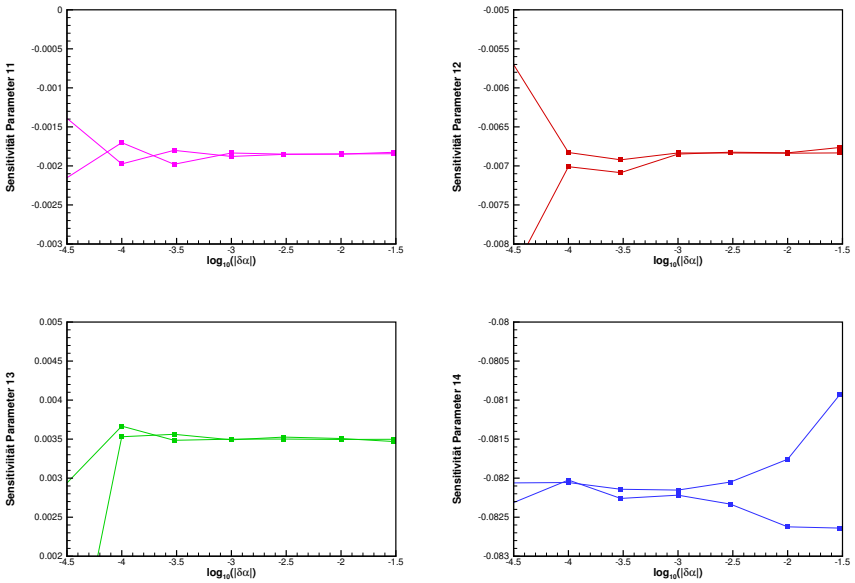


Abbildung 8.10: Studie der Deformationsweite für die finiten Differenzen der Netzerzeugungsprozesskette

Um den Einfluss der finiten Differenzen des Präprozesses auf die Genauigkeit der berechneten Sensitivitäten zu bewerten, wurde eine Parameterstudie der Deformationsweite durchgeführt. Hierzu wurden verschieden große Störungen $\delta\alpha$ auf die Eingangsparameter angewandt, um für jeden variierten Parameter $\alpha + \delta\alpha$ das zugehörige variierte Netz δx zu bilden. Die Variationen von $\delta\alpha$ lagen im Bereich 3×10^{-5} bis 3×10^{-1}. Außerdem wurde jeweils die gleiche Schrittweite in negativer Richtung angewandt, d.h. -3×10^{-5} bis -3×10^{-1}. Um eine einzelne Größe zu erhalten, wurde mit den variierten Netzen resultierende Sensitiväten des Funktionals $\mathrm{d}J/\mathrm{d}\alpha$ gebildet.

Die Ergebnisse zeigt Abbildung 8.10 für vier ausgewählte Parameter, welche sich durch unterschiedliche Verläufe auszeichnen. In den Abbildungen ist die resultierende Sensitivi-

tät logarithmisch über dem Betrag der Schrittweite aufgetragen. Dass in den Abbildungen jeweils zwei Kurven zu sehen sind, liegt daran, dass sowohl positive als auch negative Schrittweiten verwendet wurden.

Da immer die selbe adjungiert berechnete Netzsensitivität verwendet wurde, muss die Schrittweitenabhängigkeit an den finiten Differenzen des Präprozesses liegen. Hier kommen die in Abschnitt 5.2 diskutierten Probleme der Schrittweitenwahl finiter Differenzen zum Tragen.

Für jeden der dargestellten Parameter existiert ein Deformationsbereich, in dem die Abhängigkeit von der Schrittweite gering ist, jedoch ist dieser nicht bei jedem Parameter gleich und insbesondere bei Parameter 14 ist dieser Bereich auch verhältnismäßig klein. Bei der gewählten Vorgehenseise muss daher im Rahmen einer solchen Studie für jeden Parameter bestimmt werden, welche Differentiationsschrittweite zum geringsten Approximationsfehler führen. Dazu ist anzumerken, dass eine für einen bestimmten Punkt im Entwurfsraum gefundene optimale Schrittweite nicht unbedingt auf andere Punkte übertragbar sein muss. Aus diesem Grund, und weil eine solche Schrittweitenstudie aufwändig ist, wird bei jeder Vorgehensweise, bei welcher Ableitungen des Präprozess nur mittels finiter Differenzen zu erhalten sind, immer ein gewisser Fehler der Sensitivitäten zu erwarten sein. Dieser könnte durch zentrale Differenzen noch etwas reduziert werden. Für die dargestellten Parameter ist eine Genauigkeit von etwa drei signifikanten Stellen der Sensitivität zu erwarten. Diese müssen im gewählten Optimierungsverfahren als Rauschen der Gradienteninformation behandelt werden.

8.3.3 Fehlschlag von Berechnungsprozessen

Für die Optimierung ist ebenfalls relevant, dass Teile der oben beschriebenen Prozessketten auch ergebnislos abbrechen können. So kann beispielsweise die Oberflächengenerierung scheitern, da die Parametrisierung zu unmöglichen oder technisch nicht sinvollen Oberflächen führen kann. Die Netzerzeugung kann ebenfalls fehlschlagen, da die Oberflächengenerierung zu Formen führen kann, für die keine gültigen Rechennetze mehr erzeugt werden können. Ebenso kann die stationäre Strömungssimulation abbrechen, da die sich ergebenden Strömungen die Annahmen der Strömungssimulation verletzen. Dies sind bekannte Herausforderungen der Optimierung mit dreidimensionalen Strömungssimulationen. Sie sind der Grund, weshalb sich Ersatzmodellierung mit Kriging in der gradientenfreien Optimierung etabliert hat. Zuletzt kann auch die adjungierte Simulation aus den in Abschnitt 6.4.3 dargestellten Gründen fehlschlagen, so dass nur ein Teil der Ableitungen berechnet werden kann, oder diese für einen Punkt vollständig fehlen. Dies ist der wesentliche Grund, weshalb Gradient enhanced Kriging als gradientenbasierte Optimierungsmethode eingesetzt wurde.

8.3.4 Festigkeitsbedingungen

Als Festigkeitskriterium für die Rotoren wird eine Obergrenze der mechanischen Spannung gefordert. Entsprechend der Gestaltänderungsenergiehypothese (GEH) wird lokal aus dem Spannungstensor eine skalare Vergleichsspannung berechnet, die von Mises Span-

nung:

$$\sigma_v = \sqrt{\frac{1}{2}\big((\sigma_I - \sigma_{II})^2 + (\sigma_{II} - \sigma_{III})^2 + (\sigma_{III} - \sigma_I)^2\big)} \tag{8.6}$$

Diese wird aus den Hauptspannungen σ_{I-III} errechnet und es wird gefordert, dass diese im gesamten Bauteil einen materialabhängigen Grenzwert nicht überschreitet, welcher hier mit 400 MPa angesetzt wurde. Die Berechnung der Spannungen im Bauteil geschieht mittels des der Finite-Elemente Methode durch den Löser CalculiX [182]. Dabei werden die stationären aerodynamischen Druckkräfte und die Trägheitskräfte durch Rotation berücksichtigt. Der Einfachheit halber wird in dieser Arbeit ein isotropes Material angenommen. Im Original-Entwurf zu CRISPmulti wurden die anisotropen Eigenschaften des Faserverstärkten Materials berücksichtigt [183].

Das verwendete Festigkeitskriterium steht beispielhaft für die Einbindung nicht-differenzierter Bewertungskriterien. Es wird gefordert

$$\max(\sigma_{v,i}) < 400MPa \tag{8.7}$$

für alle i Elemente der Schaufeln einzeln gelten. Dies ist nicht differenzierbar, aufgrund der Sprungeigenschaft der Maximums-Funktion und der nicht-stetige Wechsel zwischen den verschiedenen Elementen welche bei kleinen Änderungen der Form den höchsten Spannungswert aufweisen.

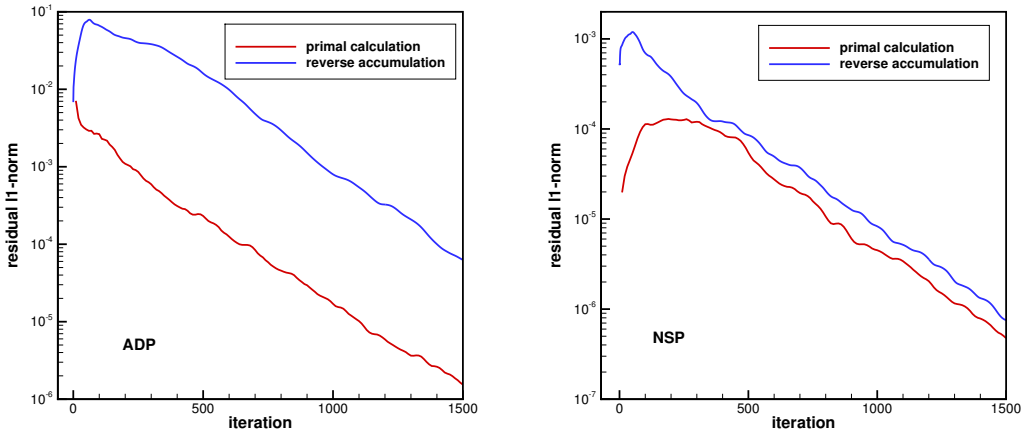
Eine der Stärken des verwendeten Kriging-Ansatzes ist die natürliche Einbindung einer solchen nicht-differenzierten Nebenbedingung in den ansonsten gradientenbasierten Optimierungsprozess. Da die Finite-Elemente Berechnung erheblich schneller mit weniger Rechenressourcen durchgeführt werden konnte als die CFD-Berechnungen wurden für die Ersatzmodelle der Festigkeitsbedingungen entsprechend mehr Simulationen durchgeführt, die normalverteilt zufällig durch Variation der kleineren Menge der CFD bewerteten Entwürfe generiert wurden.

Hierdurch konnte eine gute Vorhersage des Festigkeitsverhaltens der Schaufeln erreicht werden, so dass nur wenige Entwürfe aufgrund der tatsächlichen Festigkeitsbewertung abgelehnt wurden.

8.3.5 Konvergenzverhalten des adjungierten Lösers

Abbildung 8.11 zeigt das L1-Residuum des primalen und des adjungierten Lösers für die betrachteten Betriebspunkte ADP (Abb. 8.11a) und NSP (Abb. 8.11b). Es ist zu sehen, dass der adjungierte Löser mit der gleichen Konvergenzrate konvergiert wie der primale Löser, was für die Gültigkeit des Christianson'schen Fixpunktsatzes für diese Konfiguration spricht.

Der Konvergenzfortschritt des adjungierten Lösers wird anhand des adjungierten Residuums $\|\Delta\mu\|_1$ mit μ aus Gl. 4.36 bewertet. Interessant für die Nutzung des adjungierten Lösers in der Optimierung ist aber die Genauigkeit der Gradienten. Hierfür wäre die Veränderung der Ergebnisse $\mathrm{d}J/\mathrm{d}\alpha$ von einer Iteration zur nächsten $\|\Delta\mathrm{d}J/\mathrm{d}\alpha\|$ aussagekräftiger. Jedoch wird die Berechnung der adjungierten Variablen der Kontrolle aus Effizienzgründen nicht in der Iterationsvorschrift durchgeführt (s. Glg. 6.11) und im allgemeinen liegen während der adjungierten Simulation noch keine deformierten Netze δx



(a) ADP (b) NSP

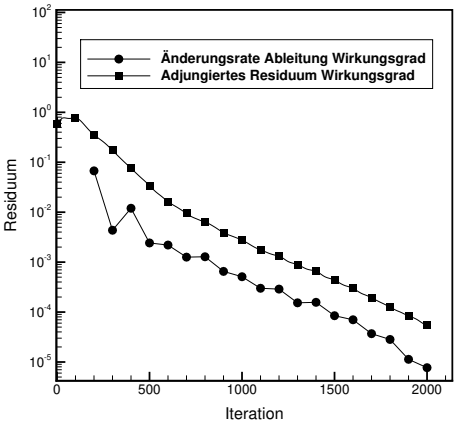
Abbildung 8.11: Konvergenz des adjungierten Lösers für die beiden Betriebspunkte (Ausgangsentwurf)

vor, um $\Delta(\mathrm{d}J/\mathrm{d}\alpha)$ auszuwerten. Es kann jedoch von der Reduktion des adjungierten Residuums $\|\Delta\mu\|_1$ auf die Verbesserung der Genauigkeit der Ergebnisse geschlossen werden. Dies ist in den Abbildungen 8.12 zu sehen. Das adjungierte Residuum $\|\Delta\mu\|_1$ wird darin gegenüber der Änderungsrate der partiellen Ableitungen von drei Funktionalen bezüglich des ersten Entwurfsparameters $\|\Delta(\mathrm{d}J/\mathrm{d}\alpha_1)\|$ dargestellt. Hierfür wurden während der adjungierten Rechnungen Zwischenergebnisse berechnet, die resultierende Ableitung bezüglich des ersten Entwurfsparameters ausgewertet $\mathrm{d}J/\mathrm{d}\alpha_1$ und die Änderung dieser Größe von einer Iteration zur nächsten dargestellt. Beide Kurven besitzen in allen drei Fällen etwa die gleiche globale Steigung, weshalb man davon ausgehen kann, dass die Reduktion des adjungierten Residuums auch etwa dem Genauigkeitsgewinn in gewünschten Ableitungen entspricht. Die Verläufe sind auch für die anderen Entwurfsparameter nahezu identisch, weshalb hier stellvertretend nur die Entwicklung für den ersten Parameter dargestellt ist.

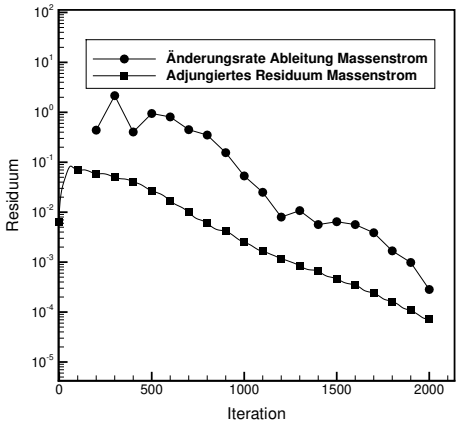
8.3.6 Ressourcenbedarf des adjungierten Verfahrens

Ein zentraler Faktor für die Effizienz des gradientenbasierten Optimierungsverfahrens ist der Ressourcenbedarf des adjungierten Lösers. Aus diesem Grund werden an dieser Stelle beide beschriebenen adjungierten Löser in Laufzeit und Speicherbedarf miteinander verglichen und in Relation zu den Ergebnissen des primalen Lösers gesetzt. Die Ergebnisse sind in Tabelle 8.3 zusammengefasst.

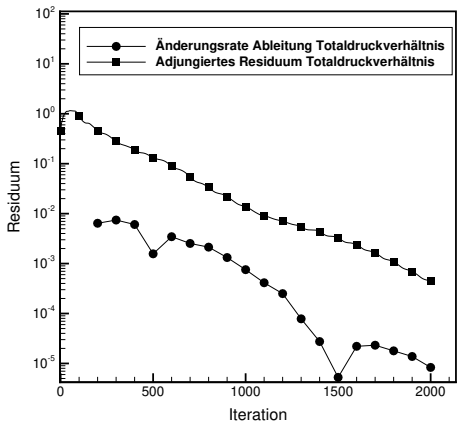
Hierbei wurde für den initialen Member die Zeit T vom Start bis zum Erreichen des eingestellten Konvergenzkriteriums gemessen. Für die Gesamtlaufzeit der verschiedenen Löser ist das eingestellte Konvergenzkriterium verantwortlich, welches nicht einheitlich definierbar ist. Daher haben die drei untersuchten Löser ihre Residuen unterschiedlich stark gesenkt (ΔR) (siehe Abb. 8.13), wobei der Unterschied mehrere Größenordnungen



(a) Isentroper Wirkungsgrad

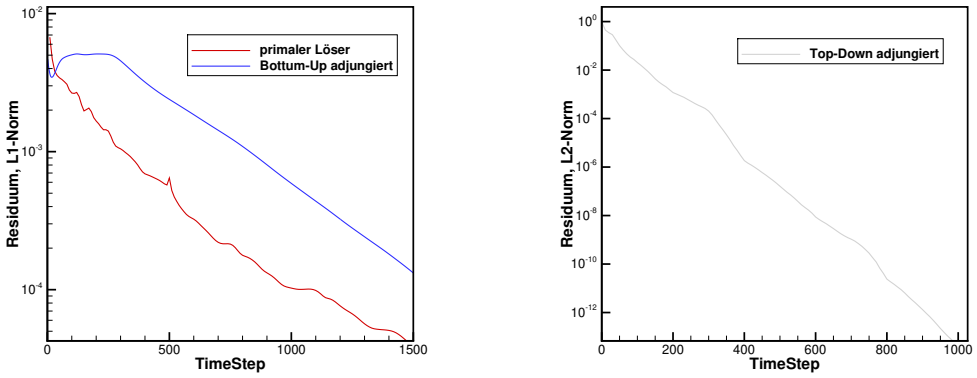


(b) Massenstrom



(c) Totaldruckverhältnis

Abbildung 8.12: Gegenüberstellung des adjungierten Residuums $\|\Delta\bar{q}\|_1$ und der Änderungsrate der Sensitivität $\|\Delta(\mathrm{d}J/\mathrm{d}\alpha)\|$ für drei Funktionale.



(a) L1-Norm des primalen und Bottom-Up adjungierten Lösers

(b) L2-Norm des Top-Down adjungierten Lösers

Abbildung 8.13: Residuenverlauf des primalen und der adjungierten Löser

beträgt. Folglich ist die benötigte Gesamtlaufzeit kein gutes Vergleichskriterium. Dabei ist anzumerken, dass für den Top-Down adjungierten Löser die L2-Norm des Residuums verwendet wird, da der darin verwendete GMRES-Löser mit dieser arbeitet, für den primalen und Bottom-Up adjungierten Löser ist hingegen die L1-Norm des Residuums als beschreibende Größe üblich. Alternativ kann man die Zeit heranziehen, welche die Verfahren benötigen, um das Residuum um eine Größenordnung zu senken ($-T/\log_{10}(\Delta R)$). Hierbei wird ein erheblicher Geschwindigkeitsunterschied zwischen den beiden adjungierten Lösern deutlich: Der Top-Down adjungierte Löser benötigt etwa 1/8 der Zeit, die der primale Löser für die Senkung des Residuums um eine Größenordnung benötigt, während der Bottom-Up adjungierte Löser hierfür etwa die dreifache Zeit des primalen Lösers benötigt.

Der Speicherbedarf des Top-Down adjungierten Verfahrens beträgt das Zehnfache des primalen Lösers bei Präkonditionierung mit SSOR und einem Restart-Intervall des GMRES-Verfahrens von 200, während der Bottom-Up adjungierte Löser ca. das Zwanzigfache des Speicherbedarfs des primalen Verfahrens benötigt. Für den Bottom-Up Löser wurden alle verfügbaren Optimierungen eingeschaltet: Die Präakkumulation der konvektiven und viskosen Flüsse, sowie die Annahme der konstanten linken Seite. Alle Ergebnisse beziehen sich auf Gleitkommaberechnungen mit doppelter Genauigkeit.

8.3.7 Verifikation der Gradienten

Entsprechend der Abwägungen aus Abschnitt 5 wird zur Verifizierung des adjungierten Lösers der Vorwärtsmodus algorithmischer Differentiation herangezogen. Durch die vollständige Überladung aller Operationen mit frei definierbaren Typen kann neben dem Black-Box Rückwärts-Löser (s. Abschnitt 4.3) auch ein Black-Box Vorwärts-Differenzierter Löser konstruiert werden. Dieser berechnet für eine gegebene Variation des Rechennetzes

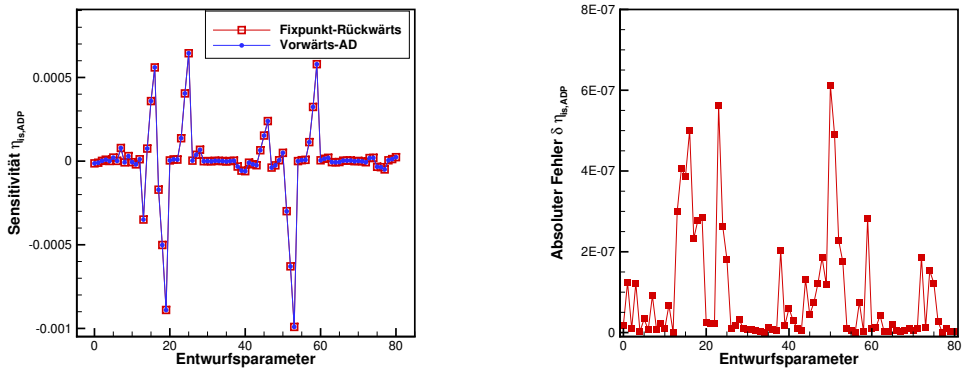
	T [s]	N	Speicher [GiB]	ΔR	$-T/\log_{10}(\Delta R)$ [s]
Primal	1550	3297	3,9	3,59E-04	450,0
Top-Down	733	1016	37,4	3,32E-14	54,4
Bottom-Up	2086	1584	76,0	3,07E-02	1379,3

Tabelle 8.3: Laufzeit- und Speicherbedarf der adjungierten Löser im Vergleich zum primalen Löser

δx die Sensitivität

$$\frac{\partial J(q_N, x)}{\partial x}\delta x. \tag{8.8}$$

Der Vorwärts-differenzierte Löser ist frei von manuell differenzierten Teilen und im Gegensatz zu finiten Differenzen frei von Schrittweiten und Auslöschungseffekten. Die berechneten Ableitungen $\partial J(q_N, x)/\partial x$ sind auf Gleitkommagenauigkeit die exakten Ableitungen für die approximative primale Lösung q_N. Lediglich nichtlineares Verhalten des Netzdeformationsprozesses kann bei der Approximation von δx mit finiten Differenzen zu einem Fehler führen, welcher in Abschnitt 8.3.2 betrachtet wurde. Abbildung 8.14a zeigt den Vergleich der Gradienten des Isentropen Wirkungsgrads bezüglich aller Entwurfsparameter im ADP des initialen Entwurfs. Hierfür wurde eine adjungierte Rechnung gegen die Ergebnisse der 81 Vorwärts-AD Rechnungen dargestellt. Da mit bloßem Auge keine Abweichung der Ergebnisse zu sehen ist, ist in Abbildung 8.14b die Differenz beider Kurven dargestellt. Es ist zu sehen, dass die Abweichung der Fixpunkt-Rückwärts-



(a) Sensitivitäten

(b) Abweichung

Abbildung 8.14: Verifikation des Gradienten der ersten Zielfunktion des initialen Designs durch Vorwärts-AD

Iteration von den Black-Box Vorwärts-Ergebnissen höchstens um 6×10^{-7} abweicht. Diese Abweichung ist für praktische Zwecke vernachlässigbar und kann durch die niemals

vollständige Konvergenz der Fixpunkt-Rückwärts-Iteration und der Finiten-Differenzen-Approximation des Prä-Prozesses erklärt werden.

8.4 Beschreibung der Optimierungen

In diesem Abschnitt der Arbeit werden die bisher beschriebenen Techniken zum Einsatz gebracht. Das Hauptziel dieser Anwendung ist es zu zeigen, dass die beschriebene Methodenkombination in einem realistischen Optimierungszenario eingesetzt werden kann.

Hierzu werden drei Optimierungen vorgestellt:

1. ohne Gradienteninformation (gradientenfrei)
2. mit adjungierten Simulationen an jedem Auswertepunkt (vollständig gradientenbasiert)
3. mit adjungierten Simulationen an jedem zehnten Auswertepunkt (teilweise gradientenbasiert)

Der Vergleich zwischen der ersten und zweiten Optimierung soll den Einfluss von Gradienteninformation auf die Ersatzmodellvorhersagen und den Optimierungsverlauf untersuchen. Die dritte Optimierung soll darstellen, dass auch bei deutlich weniger Information aus Gradienten eine ähnliche Verbesserung der Optimierung erreicht werden kann. Gute Entscheidungskriterien darüber, an welchen Punkten Gradienten berechnet werden sollen, könnten eine erhebliche Steigerung der Effizienz der Optimierung erreichen.

Ausgangsbasis jeder Optimierung sind die Ergebnisse des initialen Designs, welches alle Nebenbedingungen erfüllt. Darüber hinaus wird ein „Design of Experiments“ bestehend aus drei zufälligen Variationen dieses Designs wie folgt erzeugt: Jeder Parameter wird mit einer Wahrscheinlichkeit von 20% zur Variation ausgewählt. Ein zu variierender Parameter wird, wie in Abschnitt 7.5 beschrieben, innerhalb eines Intervalls von 10% seines Parameterbereichs verändert.

Die Optimierungsiteration besteht aus den folgenden Schritten:

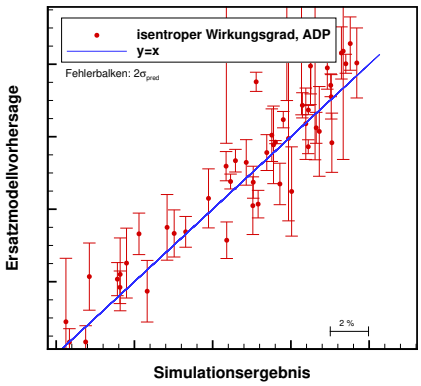
1. Entscheidung ob die Ersatzmodelle neu trainiert werden müssen, oder die Hyperparameter θ aus dem letzten Schritt genügen
2. Ggf. Training der Ersatzmodelle anhand aller vorliegenden Informationen
3. Versuche N Entwürfe durch Maximierung des erwarteten Volumenzugewinns (s. Abs. 7.7) bei gleichzeitiger Erfüllung der Nebenbedingungen durch Optimierung auf den Ersatzmodellen zu generieren
4. Ermittlung der tatsächlichen Zielfunktions- und Nebenbedingungswerte für diese N Vorschläge durch Ausführung der Auswertungsprozesskette
5. Ablegen der gewonnenen Information in einer Datenbank aller berechneten Entwürfe

Diese Iteration findet parallelisiert statt: Es können N Bewertungen gleichzeitig stattfinden und sobald eine Bewertungsprozesskette abgeschlossen ist, werden die Schritte 1 und 2 durchgeführt, um einen weiteren Vorschlag zu generieren.

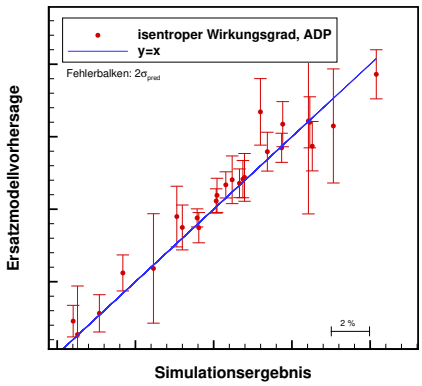
Das Hauptbewertungskriterium für die Vorhersagegüte eines Meta-Modells ist der Vorhersagefehler: der Unterschied zwischen vorhergesagtem und nachgerechnetem Wert. In Abbildung 8.15 sieht man die Vorhersage der Ersatzmodelle für beide Zielfunktionen über den Ergebnissen der Nachrechnung. Auf der x-Achse sind die zugehörigen Simulationsergebnisse aufgetragen. Bei perfekter Übereinstimmung wären x- und y-Werte gleich, diese Punkte würden auf der blauen Linie liegen. Je näher also ein Punkt der blauen Linie kommt, desto besser war der vorhergesagte Mittelwert des Ersatzmodells. Genauso wichtig wie die Mittelwertwiedergabe ist die Wiedergabe der Standardabweichung, da diese für die Berechnung des erwarteten Volumenzugewinns in der Optimierung auf dem Ersatzmodell wichtig ist. In Abbildung 8.15 ist die Standardabweichung des Vorhersagefehlers als Fehlerbalken mit Intervall 2σ dargestellt. Bei einer guten Vorhersage der Standardabweichung sollte die blaue Line von diesem Intervall eingeschlossen sein. Man kann feststellen, dass im Fall der gradientenbasierten Optimierung (Abb. 8.15b, 8.15d) die Punkte näher an der richtigen Vorhersage liegen und auch die Winkelhalbierende öfter in ihre Fehlerbalken einschließen als in den gradientenfreien Modellen (Abb. 8.15a, 8.15c). Für die gradientenfreien Modelle wird nur jede zweite Vorhersage dargestellt, um die Übersichtlichkeit zu gewährleisten.

Die Entwicklung des Vorhersagefehlers im Lauf beider Optimierungen ist in Abbildung 8.16 dargestellt. Der Fehler der gradientenfreien Ersatzmodelle ist mittels der roten, für die gradientenbasierten Ersatzmodelle als blaue Kurve dargestellt. Es ist zu sehen, dass der gradientenbasierte Vorhersagefehler bereits nach Abschluss des DOE um eine Größenordnung kleiner ist, als im gradientenfreien Fall. Es ist zu beachten, dass das DOE in den Abbildungen 8.16 nicht dargestellt ist. In beiden Fällen umfasst es die gleichen drei Evaluationen zufälliger Variationen des initialen Entwurfs. Die gradientenfreien Ersatzmodelle erreichen auch nach 130 Optimierungsiterationen nur einen höheren Vorhersagefehler. Dieses Ergebnis entspricht insofern den Erwartungen, als dass die gradientenbasierten Modelle mit einer Korrelationsmatrix der Größe 324×324 starten. Die Größe dieser Matrix ergibt sich aus $(1+n_{\text{parameter}})n_{\text{samples}}$ mit $n_{\text{parameter}} = 81$ und $n_{\text{samples}} = 3$, da für alle drei Variationen Funktionswerte und Gradienten berechnet wurden. Diese Informationsmenge ist offensichtlich ausreichend, um eine bessere Vorhersagequalität als nach etwa 130 primalen Zielfunktionsauswertungen zu ermöglichen. Dass der Informationsgewinn durch die Gradienten nicht vollständig äquivalent zum Informationsgewinn durch Funktionswerte ist, wird auch durch die Beobachtungen in Abschnitt 7.4 gestützt, in dem die Vorhersagegenauigkeit der Funktion mit 8 Funktionswerten besser ist, als mit 4 Funktionswerten und 4 partiellen Ableitungen.

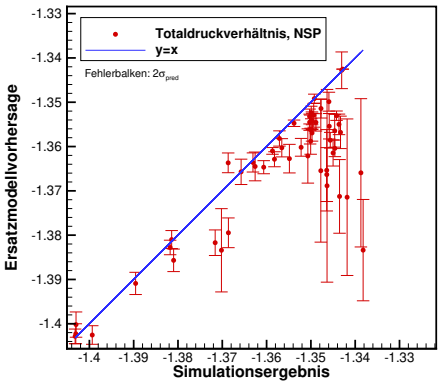
Weiterhin kann beobachtet werden, dass der Vorhersagefehler im Laufe der Optimierung nicht weiter sinkt, obwohl weitere Datenpunkte hinzugefügt werden. Dies kann man so interpretieren, dass die Informationsmenge am Anfang schon ausreichend ist, um eine hinreichend gute Wiedergabe der Modelle zu ermöglichen. Da weitere Information hier keinen Gewinn an Vorhersagegenauigkeit bietet, motiviert diese Beobachtung, die Zahl der Gradientenauswertungen zu reduzieren. Mit sinkender Korrelation neu vorhergesag-



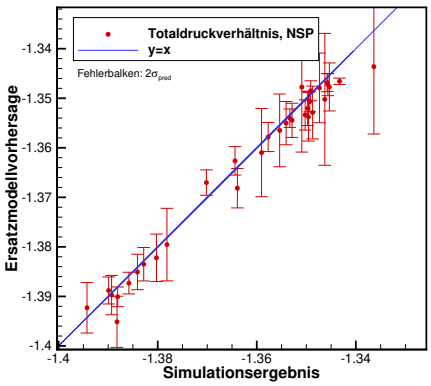
(a) Erste Zielfunktion, gradientenfrei



(b) Erste Zielfunktion, gradientenbasiert

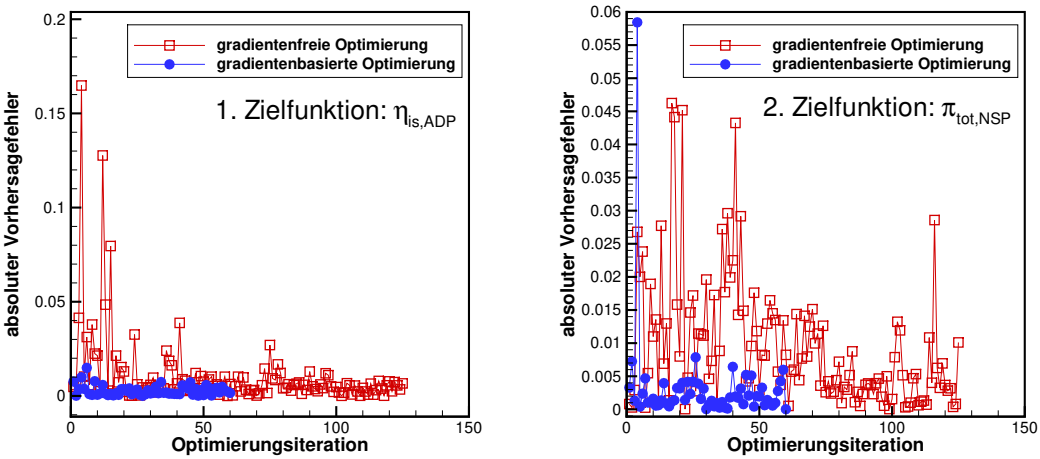


(c) Zweite Zielfunktion, gradientenfrei



(d) Zweite Zielfunktion gradientenbasiert

Abbildung 8.15: Einfluss der Gradienteninformation auf die Vorhersagegüte für Mittelwert und Standardabweichung im Vergleich zu nachgerechneten Werten für beide Zielfunktionen.



(a) Erste Zielfunktion

(b) Zweite Zielfunktion

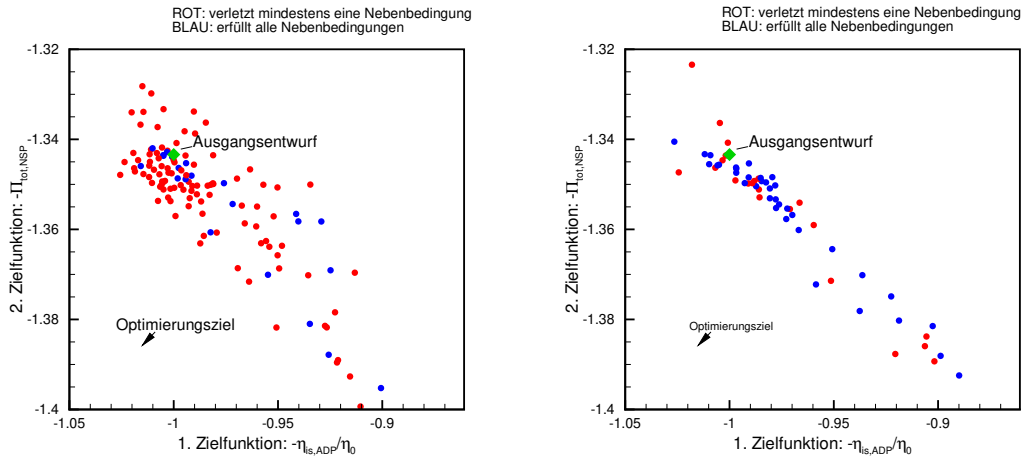
Abbildung 8.16: Entwicklung des Vorhersagefehlers im Verlauf der Optimierungen für beide Zielfunktionen im Vergleich zwischen gradientenfreiem und gradientenbasiertem Modell.

ter Designs mit bereits berechneten Punkten, kann es dennoch sinnvoll sein, Information aus adjungierten Rechnungen zum Modell hinzuzufügen.

In Abbildung 8.17 sind die resultierenden Pareto-Fronten aus der gradientenfreien und der vollständig gradientenbasierten Optimierung im Zielfunktionsraum dargestellt. Beide Zielfunktionen sollen maximiert werden. Da der Optimierer als Minimierer konzipiert ist, sind die Zielfunktionswerte negiert. Das Optimum befindet sich demnach in der linken unteren Ecke der Abbildungen. Jeder dargestellte Punkt entspricht einer erfolgreichen Auswertung aller Zielfunktionen und Nebenbedingungen. Verletzt ein Punkt eine der Nebenbedingungen, kann er nach Definition nicht mehr Pareto-Optimal sein, und ist daher rot dargestellt. Punkte, welche alle Nebenbedingungen erfüllen sind blau dargestellt. Aus beiden Bildern ist ersichtlich, dass die Pareto-Front nahezu linear verläuft, was ein Zeichen der Gegensätzlichkeit beider Ziele ist. In der vollständig gradientenbasierten Optimierung ist der Anteil der Punkte, welche alle geforderten Nebenbedingungen erfüllen, wesentlich höher. Darüber hinaus ist die Streuung der Punkte deutlich niedriger. Beide Phänomene sind direkte Konsequenzen aus der verbesserten Vorhersage durch die Ersatzmodelle. In beiden Abbildungen sind Punkte zu sehen, welche einen Volumenzugewinn in der jeweils anderen Optimierung bedeuten würden. Dies ist eine Folge daraus, dass beide Optimierungen noch nicht auskonvergiert sind und somit der Einfluss des Zufalls auf die Optimierung auf den Ersatzmodellen sichtbar wird.

Der Fortschritt der Optimierungen kann gut anhand des kumulativen Volumenzugewinns zur Pareto-Front bewertet werden, wie in Abschnitt 7.7 beschrieben wurde.

Dieses Maß ist in Abbildung 8.18a für die Optimierungen über der Optimierungsite-



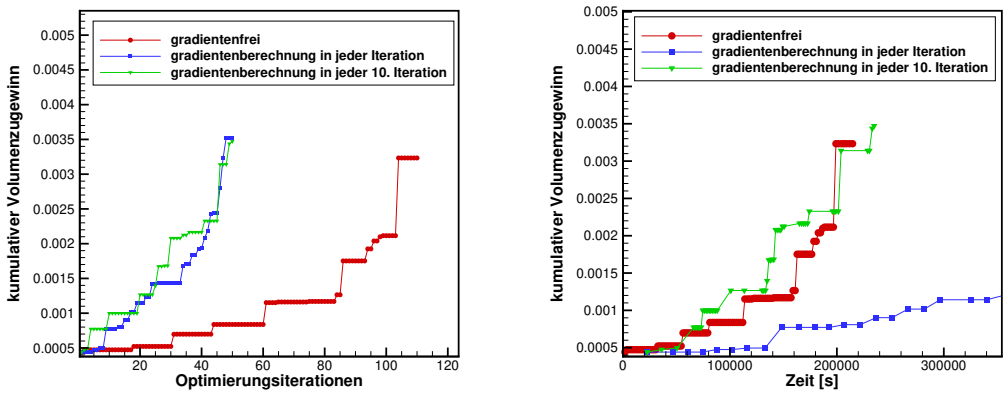
(a) Gradientenfreie Optimierung (b) Gradientenbasierte Optimierung

Abbildung 8.17: Vergleich der Pareto-Fronten aus der gradientenfreien und gradientenbasierten Optimierung

ration dargestellt. Die Steigung der Kurven ist somit ein Maß für den Fortschritt der Optimierung pro Iteration.

Beide gradientenbasierte Optimierungen steigen etwa gleich schnell an und tun dies mit einer höheren Steigungsrate als die gradientenfreie Optimierung. In allen Kurven sieht man mehr oder weniger ausgeprägte Plateaus in denen kein Optimierungsfortschritt erreicht wird. Sie stammen aus Bewertungen, deren Pareto-Rang (s. Abschnitt 2.3) größer als eins ist oder die Aufgrund einer verletzten Restriktion nicht zulässig sind. Diese Plateaus sind ebenfalls in der gradientenfreien Optimierungen ausgeprägter als in beiden gradientenbasierten Optimierungen. Beide gradientenbasierte Optimierungen bieten einen etwa gleich schnellen Fortschritt, obwohl in der teilweise gradientenbasierten Optimierung nur ein Zehntel der Gradienteninformation zur Verfügung steht. Dass die teilweise gradientenbasierte Optimierung teilweise einen steileren Anstieg aufweist, kann durch die stochastische Natur der Optimierung auf dem Ersatzmodell erklärt werden. Die nahezu gleiche Steigung beider gradientenbasierten Optimierungen bekräftigt die Vermutung, dass hier eine kleine Anzahl Gradienten ausreichend ist, um eine gute Vorhersage der Zielfunktion zu erreichen. Die Betrachtung guter Entscheidungskriterien zur selektiven Berechnung von Gradienten in der Ersatzmodellierung erscheint hierdurch als aussichtsreiches Forschungsthema.

Vergleicht man den Optimierungsfortschritt über der Zeit wie in Abbildung 8.18b, sieht man, dass die Erweiterung des Kriging Modells mit Gradienteninformation in jedem Punkt nicht effizient ist. Die zusätzliche Gradientenberechnung in jedem Schritt benötigt so viel Zeit, dass diese Optimierung sogar langsamer voranschreitet, als die gradientenfreie Optimierung. Hier ist zu vermuten, dass eine zu hohe Informationsdichte vorliegt, wie sie auch im Experiment zur Kriging Konditionszahl in Abschnitt 7.4 auftrat, wenn für alle Punkte Gradienten in das Modell integriert wurden. Ab einer gewissen Informationsdichte



(a) Vergleich per Iteration

(b) Vergleich per Zeit

Abbildung 8.18: Vergleich des Optimierungsfortschritts

erhöhen die Gradienten die Wiedergabegenauigkeit des Modells nicht weiter, vielmehr erschweren sie das Training, da sie die Konditionszahl erheblich steigern.

Diese Beobachtung ist der Grund, weshalb eine Optimierung mit Gradientenberechnung in nur jeder 10. Iteration durchgeführt wurde. Da sie pro Iteration etwa genau so schnell voranschreitet wie die vollständig gradientenbasierte Optimierung, reduziert sich bei ihr der Rechenaufwand für die Gradientenberechnung auf ein Zehntel, bei gleichwertiger Ersatzmodellgenauigkeit, was dazu führt, dass die gradientenbasierte Berechnung im Zeitaufwand gleichauf mit der gradientenfreien Rechnung wird.

Als Resultat der gradientenbasierten Optimierung wurde hier das Design mit dem höchsten Wirkungsgrad ausgewählt. In Abbildung 8.19 sieht man den Vergleich der entstandenen Konstruktionsprofile und dazwischen interpolierte Profile von Ausgangsdesign und des optimierten Designs.

8.5 Optimierungsresultat

Die Erreichung der gestellten Zielvorgaben ist am besten im Vergleich der Arbeitslinien zu erkennen. Dieser ist in Abbildung 8.20 dargestellt. In der Totaldruck–Kennlinie (Abb. 8.20a) ist zu sehen, dass die Beschränkungen bezüglich des maximal zulässigen Massenstroms im ADP ebenso wie die Untergrenzen von Massenstrom und Totaldruckverhältnis im NSP maßgebliche Beschränkungen für die Steigerung des Wirkungsgrades im ADP (siehe Abb. 8.20b) darstellen, da die entsprechenden Grenzen der Nebenbedingungen für die Kennlinie des Members mit höchstem Wirkungsgrad exakt getroffen werden. Der Betriebsbereich der Konfiguration mit optimalem Wirkungsgrad ist ebenfalls, wie durch die entsprechende Nebenbedingung gefordert, nicht verkleinert worden.

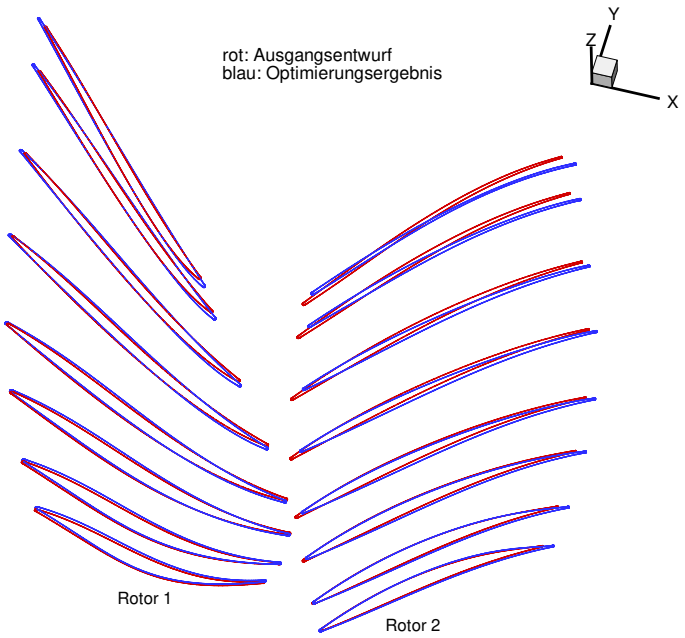
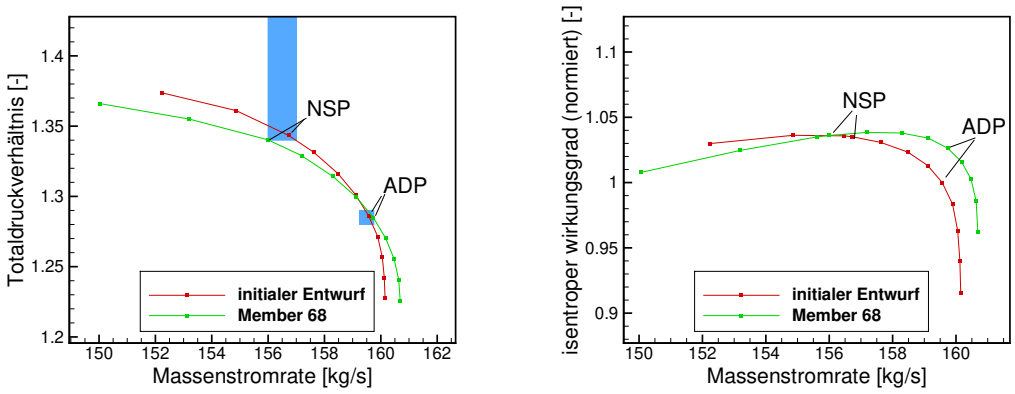


Abbildung 8.19: Vergleich der Konstruktionsprofile zwischen initialem und gradientenbasiert optimierten Design.



(a) Totaldruckverhältnis

(b) Isentroper Wirkungsgrad

Abbildung 8.20: Vergleich der Kennlinien zwischen initialem und gradientebasiert optimiertem Design

9 Zusammenfassung

Um die vorhandenen Optimierungspotentiale von Turbomaschinen auszuschöpfen, wird die Zahl der Freiheitsgrade in numerischen Auslegungsoptimierungen kontinuierlich gesteigert. Die daraus resultierende Zahl zur Bewertung der Entwürfe benötigter Simulationen wächst jedoch schneller als die verfügbare Rechenkapazität. Adjungiert berechnete Gradienten des Optimierungsproblems sind ein vielversprechender Lösungsansatz für dieses Dilemma. Sie liefern eine lokale Optimierungsrichtung zu Kosten die nicht von der Zahl der Freiheitsgrade abhängen, sondern nur von der Zahl der Zielfunktionen und Nebenbedingungen. In dieser Arbeit werden die Herausforderungen diskutiert die bei der Berechnung und Verwendung adjungierter Gradienten in multidisziplinären Turbomaschinenauslegungen auftreten.

Zuerst wird der Einfluss der Vorgehensweise zur Implementierung auf verschiedene Aspekte der Einsetzbarkeit diskutiert. Es werden zwei Vorgehensmodelle zur Erstellung eines diskret adjungierten Lösers vorgestellt: Top-Down (manuelle adjungierung mit algorithmischer Differentiation zugrundeliegender Routinen) und Bottom-Up (algorithmische Differentiation des gesamten Lösers und anschließende manuelle Optimierungen). Zu beiden Ansätzen wird die Realisierung für den gleichen zugrundeliegenden primalen Löser vorgestellt. Der primale Löser unterliegt einer stetigen Veränderung. Es werden Verfahren und Modelle durch mehrere, räumlich verteilte Entwicklergruppen, neu hinzugefügt oder verbessert. In der Top-Down Vorgehensweise ist es sehr schwer zu erkennen, ob eine manuell implementierte Adjungierung noch zum primalen Löser passt. Falls dies nicht der Fall ist, muss Entwicklungsaufwand investiert werden um ein geändertes Modell erneut zu adjungieren. Noch mehr Aufwand ist nötig ein neu hinzugefügtes Modell erstmalig zu adjungieren. Aus diesen Gründen ist der Bottom-Up Ansatz vorteilhaft, da er eine Verifizierung der Konsistenz beider Verfahren erlaubt. Der optimierte adjungierte Löser kann stets gegen ausschließlich algorithmisch differenzierte Verfahren verglichen werden. Neue Modelle sind schneller verfügbar, manueller Aufwand wird eventuell zur Beschleunigung und Speicherreduktion fällig. Der Bottom-Up Ansatz führt zuerst jedoch zu einem adjungierten Verfahren mit erhöhtem Speicher- und Rechenzeitbedarf verglichen mit der Top-Down Vorgehensweise. Mit gezielten Maßnahmen wird jedoch in der Bottom-Up Vorgehensweise durch Ausnutzung von Struktur und Algorithmen des primalen Verfahrens der Ressourcenbedarf erheblich gesenkt. Da man von einem verifizierten Verfahren ausgeht, kann sichergestellt werden, dass die Verbesserungen die Ergebnisse des Verfahrens nicht beeinflussen. Diese Kette der Verifizierbarkeit ist der wesentliche Vorteil des Bottom-Up Ansatzes zur Adjungierung eines existierenden CFD-Verfahrens.

Der zweite Teil der Arbeit befasst sich mit der Einbindung der gewonnenen Gradienten in multidisziplinäre Mehrzieloptimierungen aus der Auslegungspraxis. Besonderes Augenmerk gilt dabei der Zusammenstellung der adjungierten Prozesskette zur Gradien-

tenberechnung. Die aus der bereits etablierten gradientenfreien Optimierung bekannten Kriging-Ersatzmodelle werden in Form direkt gradientenerweitertem Krigings eingesetzt. Dies hat sich gegenüber Gradientenabstiegverfahren als vorteilhaft erwiesen, vor allem aus Gründen der Fehlertoleranz, Verwendbarkeit nicht-differenzierter Bewertungsfunktionen, Fähigkeit zur Überwindung lokaler Minima sowie kontrollierbare Glättungseigenschaften für numerisch bedingte Unsicherheiten der Zielfunktionsauswertung.

Die beschriebenen Techniken werden im Zusammenspiel an der Optimierung des gegenläufigen Triebwerksfans CRISPmulti angewandt. Hierdurch wird gezeigt, dass die Gesamtheit der vorgestellten numerischen Verfahren miteinander in einer robusten Optimierungsprozesskette integriert werden kann.

Danksagung

Ich danke dem BMWi für die Förderung meiner Arbeit durch das R&E-Turb, Luftfahrtforschungsprogramms IV sowie dem EU Rahmenprogramms CleanSky 2 für die Förderung im Förderprogramm Horizon 2020 Projekt 2-Shaft-Compressor unter der Förderkennung Horizon 2020 CS2-Engines ITD-2014-2015-01.

Für die Übernahme der Betreuung, die Begleitung durch fachliche Diskussionen, Nachfragen zum Stand der Arbeit und die schnelle Erreichbarkeit für alle Fragen die ich im Laufe dieser Promotion hatte, gilt mein Dank meinem Doktorvater Professor Mönig und meinem zweiten Betreuer Professor Gottschalk.

Meinem Abteilungsleiter Edmund Kügeler verdanke ich, für die Förderung und dass meine Arbeit für das DLR im Einklang mit meinem Dissertationsthema stand und sich so für mich, aber auch die von mir bearbeiteten Projekte, viele fachliche Synergien ergeben haben. Auch die Rückmeldungen zu Text-Entwürfen, die fachliche Betreuung zur Turbomaschinen-CFD und die organisatorische Unterstützung waren sehr wertvolle Hilfen für mich.

Für die fachliche Betreuung von der Themenfindung bis zur fertig geschriebenen Dissertation, durch Fachdiskussionen, Richtungsimpulse und Korrekturlesen und gilt mein Dank Christian Frey.

Mein Dank gilt auch Anna Engels-Putzka für die Unterstützung bei der Entwicklung und Fehlersuche in adjungierten Lösern, Prozessketten und mathematische Überlegungen, aber auch für gemeinsame Veröffentlichungen und das Korrekturlesen der ersten Entwürfe dieser Arbeit.

Bei den Themen Kriging und Optimierung habe ich eng mit Kollegen aus der Abteilung Fan und Verdichter zusammengearbeitet. Für die tolle Zusammenarbeit und Unterstützung zu gradientenbasiertem Kriging und Optimierungsverfahren danke ich ganz besonders Christian Voß, Andreas Schmitz und Marcel Aulich.

Der Anwendungsfall meiner Arbeit basiert auf CRISPmulti, wofür ich von Timea Lengyel und Anne-Laure LeDenmat die Aerodynamische Bewertungskette als Grundlage für den adjungierten Prozess übernehmen durfte und mich mit allen Fragen zum Design an sie wenden durfte.

Von der ersten Vorstellung im Institut für Antriebstechnik an, hat Eberhard Nicke mich gefördert und unterstützt, wofür ich ihm sehr dankbar bin.

Ich möchte auch insgesamt allen Kollegen aus dem Institut für Antriebstechnik und im besonderen allen Abteilungskollegen der numerischen Methoden und den Kollegen von Fan und Verdichter danken. Es macht mir Spaß mit so vielen motivierten Menschen mit so vielfältigem Fachwissen, Interessen, und Fähigkeiten zusammenarbeiten zu dürfen.

Dem DLR danke ich als Arbeitgeber, der beispielsweise durch das Graduate Program mich und meine Promotion sehr gefördert hat.

Der Löser adAdjointTRACE ist in einer Kooperation zwischen TU-Kaiserslautern, MTU und DLR hervorgegangen. Für die sehr enge und produktive Zusammenarbeit an diesem herausfordernden Projekt danke ich Max Sagebaum und Sebastian Mann - es hat mir viel Freude gemacht Teil dieses Teams zu sein.

Ich danke Christian, Jens, Dirk, Kai und Christian für die Gelegenheiten sich über das Vorankommen der Promotion auch ganz Abseits des Arbeitsplatzes austauschen zu können.

Meine Familie und meine Freunde haben mir den Rückhalt gegeben, den ich zur Erstellung dieser Arbeit gebraucht habe. Danke!

Literaturverzeichnis

[1] J. L. Lions. *Optimal control of systems governed by partial differential equations*. Springer, 1971.

[2] O. Pironneau. On optimum design in fluid mechanics. *Journal of Fluid Mechanics*, 64(01):97–110, 1974.

[3] A. Jameson. Aerodynamic design via control theory. *Journal of scientific computing*, 3(3):233–260, 1988.

[4] A. Jameson, L. Martinelli, and N. Pierce. Optimum aerodynamic design using the Navier–Stokes equations. *Theoretical and computational fluid dynamics*, 10(1-4):213–237, 1998.

[5] A. Jameson. Optimum aerodynamic design using CFD and control theory. *AIAA paper*, 1729:124–131, 1995.

[6] B. Mohammadi and O. Pironneau. *Applied shape optimization for fluids*. Oxford university press, 2010.

[7] J. E. Peter and R. P. Dwight. Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches. *Computers & Fluids*, 39(3):373–391, 2010.

[8] M. B. Giles and N. A. Pierce. Adjoint equations in CFD: duality, boundary conditions and solution behaviour. *AIAA paper*, 97:1850, 1997.

[9] S. Nadarajah and A. Jameson. A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization. *AIAA paper*, 667:2000, 2000.

[10] D. Papadimitriou, A. Zymaris, and K. Giannakoglou. Discrete and continuous adjoint formulations for turbomachinery applications. In *EUROGEN 2005, international conference proceedings, Munich*, 2005.

[11] M. B. Giles and N. A. Pierce. An introduction to the adjoint approach to design. *Flow, turbulence and combustion*, 65(3-4):393–415, 2000.

[12] J. R. R. A. Martins and J. T. Hwang. Review and unification of methods for computing derivatives of multidisciplinary computational models. *AIAA Journal*, 51(11):2582–2599, November 2013.

[13] J.-D. Müller and P. Cusdin. On the performance of discrete adjoint CFD codes using automatic differentiation. *Int. J. Numer. Meth. Fluids*, 47:939–945, January 2005.

[14] C. Frey, H.-P. Kersken, and D. Nürnberger. The discrete adjoint of a turbomachinery RANS solver. In *ASME Turbo Expo 2009*, pages 345–354. American Society of Mechanical Engineers, 2009.

[15] C. E. Burdyshaw and W. K. Anderson. A general and extensible unstructured mesh adjoint method. *Journal of Aerospace Computing, Information, and Communication*, 2(10):401–413, 2005.

[16] E. J. Nielsen and W. L. Kleb. Efficient construction of discrete adjoint operators on unstructured grids using complex variables. *AIAA journal*, 44(4):827–836, 2006.

[17] A. Griewank and A. Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. Frontiers in Applied Mathematics. SIAM, 2008.

[18] U. Naumann. *The art of differentiating computer programs: an introduction to algorithmic differentiation*, volume 24 of *Software, Environments, Tool*. SIAM, 2012.

[19] R. P. Dwight and J. Brezillon. Effect of approximations of the discrete adjoint on gradient-based optimization. *AIAA journal*, 44(12):3022–3031, 2006.

[20] B. Mohammadi, J. Malé, and N. Rostaing-Schmidt. Automatic differentiation in direct and reverse modes: application to optimum shapes design in fluid mechanics. *Computational Differentiation: Techniques, Applications, and Tools*, pages 309–318, 1996.

[21] M. Towara and U. Naumann. A discrete adjoint model for OpenFOAM. *Procedia Computer Science*, 18(0):429 – 438, 2013. 2013 International Conference on Computational Science.

[22] N. Kroll, K. Becker, H. Rieger, and F. Thiele. Ongoing activities in flow simulation and shape optimization within the german megadesign project. In *25th International Congress of the Aeronautical Sciences, ICAS*, 2006.

[23] M. B. Giles, D. Ghate, and M. C. Duta. Using automatic differentiation for adjoint CFD code development. Technical report, University of Oxford, Eprints Archive, 2005.

[24] C. A. Mader, J. RA Martins, J. J. Alonso, and E. V. Der Weide. ADjoint: An approach for the rapid development of discrete adjoint solvers. *AIAA journal*, 46(4):863–873, 2008.

[25] A. C. Marta, S. Shankaran, D. G. Holmes, and A. Stein. Development of adjoint solvers for engineering gradient-based turbomachinery design applications. In *ASME Turbo Expo 2009*, volume 7: Turbomachinery, Parts A and B, June 2009.

[26] T. Albring, M. Sagebaum, and N. R. Gauger. Development of a consistent discrete adjoint solver in an evolving aerodynamic design framework. In *16th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference. American Institute of Aeronautics and Astronautics*, 2015.

[27] M. Sagebaum, E. Özkaya, and N. R. Gauger. Challenges in the automatic differentiation of an industrial CFD solver. In *Evolutionary and Deterministic Methods for Design, Optimization and Control with Application to Industrial and Societal Problems (EUROGEN 2013)*, 2013.

[28] J. Backhaus, A. Engels-Putzka, and C. Frey. A code-coupling approach to the implementation of discrete adjoint solvers based on automatic differentiation. In *VII European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS)*, June 2016.

[29] M. Sagebaum, E. Özkaya, N. R. Gauger, J. Backhaus, C. Frey, S. Mann, and M. Nagel. Efficient Algorithmic Differentiation Techniques for Turbo-machinery Design. *AIAA-Paper AIAA-2017-3998*, (AIAA-2017-3998), 2017.

[30] S. J. Leary, A. Bhaskar, and A. J. Keane. Global approximation and optimization using adjoint computational fluid dynamics codes. *AIAA journal*, 42(3):631–641, 2004.

[31] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423, Nov. 1989.

[32] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[33] J. Peter and M. Marcelet. Comparison of surrogate models for turbomachinery design. *WSEAS Transactions on Fluid Mechanics*, 3(1):10–17, 2008.

[34] M. Aulich and U. Siller. High-dimensional constrained multiobjective optimization of a fan stage. In *ASME 2011 Turbo Expo*, pages 1185–1196. American Society of Mechanical Engineers, 2011.

[35] H.-S. Chung and J. J. Alonso. Using gradients to construct response surface models for high-dimensional design optimization problems. *AIAA Paper*, 922:2001, 2001.

[36] H.-S. Chung and J. J. Alonso. Using gradients to construct cokriging approximation models for high-dimensional design optimization problems. *AIAA Paper*, 317:14–17, 2002.

[37] M. D. Morris, T. J. Mitchell, and D. Ylvisaker. Bayesian design and analysis of computer experiments: use of derivatives in surface prediction. *Technometrics*, 35(3):243–255, 1993.

[38] A. Forrester, A. Sóbester, and A. Keane. *Engineering Design via Surrogate Modelling: A Practical Guide*. Wiley, 2008.

[39] K. R. Dalbey. Efficient and robust gradient enhanced kriging emulators. SANDIA REPORT SAND2013-7022, Sandia Natianal Laboratories, Sandia National Laboratories Albuquerque, New Mexico 87185 and Livermore, California 94550, August 2013.

[40] R. Zimmermann. On the maximum likelihood training of gradient-enhanced spatial gaussian processes. *SIAM Journal on Scientific Computing*, 35(6):A2554–A2574, 2013.

[41] Z.-H. Han, S. Görtz, and R. Zimmermann. Improving adjoint-based aerodynamic optimization via gradient-enhanced kriging. In *50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition, Nashville, Tennessee*, 2012.

[42] M. S. Campobasso, M. C. Duta, and M. B. Giles. Adjoint calculation of sensitivities of turbomachinery objective functions. *Journal of propulsion and power*, 19(4):693–703, 2003.

[43] H. Wu, F. Liu, and H. Tsai. Aerodynamic design of turbine blades using an adjoint equation method. *AIAA paper*, 1006:10–13, 2005.

[44] D. Papadimitriou and K. Giannakoglou. Total pressure loss minimization in turbomachinery cascades using a new continuous adjoint formulation. *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, 221(6):865–872, 2007.

[45] M. C. Duta, S. Shahpar, and M. B. Giles. Turbomachinery design optimization using automatic differentiated adjoint code. In *ASME Turbo Expo 2007*, pages 1435–1444. American Society of Mechanical Engineers, 2007.

[46] S. Shahpar and S. Caloni. Adjoint optimisation of a high-pressure turbine stage for a lean-burn combustion system. *Proceeding of 10th ETC, Lappeenranta, Finland*, 2013.

[47] M. Hilditch, A. Fowler, T. Jones, K. Chana, M. Oldfield, R. Ainsworth, S. Hogg, S. Anderson, and G. Smith. Installation of a turbine stage in the pyestock isentropic light piston facility. In *ASME 1994 International Gas Turbine and Aeroengine Congress and Exposition*, pages V004T09A043–V004T09A043. American Society of Mechanical Engineers, 1994.

[48] B. Walther and S. Nadarajah. Optimum shape design for multirow turbomachinery configurations using a discrete adjoint approach and an efficient radial basis function deformation scheme for complex multiblock grids. *Journal of Turbomachinery*, 137(8):081006, 2015.

[49] G. Schulze, D. K. Hennecke, J. Sieber, and B. Wöhrl. Der neue Verdichterprüfstand an der TH Darmstadt. *VDI Berichte Nr. 1109, Germany.*, 1994.

[50] C. Blaha, D. Hennecke, G. Fritsch, M. Höger, and M. Beversdorff. Laser-2-focus measurements in a transonic compressor blisk-rotor and comparison with 3 d numerical simulations. In *ISABE- International Symposium on Air Breathing Engines, 13 th, Chattanooga, TN*, pages 484–493, 1997.

[51] D. X. Wang and L. He. Adjoint aerodynamic design optimization for blades in multistage turbomachines—part I: Methodology and verification. *Journal of Turbomachinery*, 132(2):021011, 2010.

[52] D. X. Wang, L. He, Y. S. Li, and R. G. Wells. Adjoint aerodynamic design optimization for blades in multistage turbomachines—part II: Validation and application. *Journal of Turbomachinery*, 132(2):021012, 2010.

[53] J. Backhaus, M. Aulich, C. Frey, T. Lengyel, and C. Voß. Gradient enhanced surrogate models based on adjoint CFD methods for the design of a counter rotating turbofan. In *ASME Turbo Expo 2012*, 2012.

[54] C. S. Kim, C. Kim, and O. H. Rho. Feasibility study of constant eddy-viscosity assumption in gradient-based design optimization. *J. Aircraft*, 40(6):1168–1176, 2003.

[55] A. C. Marta and S. Shankaran. On the handling of turbulence equations in RANS adjoint solvers. *Computers & Fluids*, 74:102–113, 2013.

[56] H.-J. Kim, C. Kim, O.-H. Rho, and K. D. Lee. Aerodynamic sensitivity analysis for navier-stokes equations. In *Proceedings of the 37th AIAA Aerospace Sciences Meeting*. Korea Society for Industrial and Applied Mathematics, 1999.

[57] E. J. Nielsen and W. K. Anderson. Aerodynamic design optimization on unstructured meshes using the navier-stokes equations. *AIAA journal*, 37(11), 1999.

[58] M. B. Giles, M. C. Duta, and J.-D. Müller. Adjoint code developments using the exact discrete approach. *ROLLS ROYCE PLC-REPORT-PNR*, 2001.

[59] I. Kavvadias, E. Papoutsis-Kiachagias, G. Dimitrakopoulos, and K. Giannakoglou. The continuous adjoint approach to the k–ω sst turbulence model with applications in shape optimization. *Engineering Optimization*, 47(11):1523–1542, 2015.

[60] J.-D. Lee and A. Jameson. Natural-laminar-flow airfoil and wing design by adjoint method and automatic transition prediction. In *Proceedings of the 47th AIAA Aerospace Sciences Meeting*, pages 1–25, 2009.

[61] P. Khayatzadeh and S. K. Nadarajah. Aerodynamic shape optimization via discrete viscous adjoint equations for the k- ωsst turbulence and γ- reθ transition models. In *49th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition*, 2011.

[62] A. Giebmanns, J. Backhaus, C. Frey, and R. Schnell. Compressor leading edge sensitivities and analysis with an adjoint flow solver. In *ASME Turbo Expo 2013*, number GT2013-94427, 2013.

[63] A. Giebmanns, R. Schnell, and C. Werner-Spatz. A method for efficient performance prediction for fan and compressor stages with degraded blades. In *ASME Turbo Expo 2015*, June 2015.

[64] A. Giebmanns. *Effiziente Bestimmung der Kennfeldsensitivitäten von Fans und Verdichtern bei kleinen Änderungen der Schaufelgeometrie.* PhD thesis, Ruhr-Universität Bochum, 2017.

[65] G. Zamboni, G. Banks, and S. Bather. Gradient-based adjoint and design of experiment CFD methodologies to improve the manufacturability of high pressure turbine blades. In *ASME Turbo Expo 2016*, pages V02CT39A001–V02CT39A001. American Society of Mechanical Engineers, 2016.

[66] A. Liefke, V. Marciniak, U. Janoske, and H. Gottschalk. Using adjoint CFD to quantify the impact of manufacturing variations on a heavy duty turbine vane. In *European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS)*, 2018.

[67] A. Liefke, V. Marciniak, J. Backhaus, C. Frey, H. Gottschalk, and U. Janoske. Aerodynamic impact of manufacturing variation on a nonaxisymmetric multi-passage turbine stage with adjoint CFD. In *ASME Turbo Expo 2019*, 2019.

[68] F. Montomoli, M. Carnevale, A. D'Ammaro, M. Massini, and S. Salvadori. *Uncertainty quantification in computational fluid dynamics and aircraft engines.* Springer, 2015.

[69] M. Massini and F. Montomoli. Manufacturing/in-service uncertainty and impact on life and performance of gas turbines/aircraft engines. In *Uncertainty Quantification in Computational Fluid Dynamics and Aircraft Engines*, pages 1–32. Springer, 2019.

[70] R. P. Dwight and Z.-H. Han. Efficient uncertainty quantification using gradient-enhanced kriging. *AIAA paper*, 2276:2009, 2009.

[71] A. Engels-Putzka, J. Backhaus, and C. Frey. On the usage of finite differences for the development of discrete linearised and adjoint CFD solvers. In E. Oñate, X. Oliver, and A. Huerta, editors, *VI European Conference on Computational Fluid Dynamics (ECFD)*, Juli 2014.

[72] C. Voss, M. Aulich, B. Kaplan, and E. Nicke. Automated multiobjective optimisation in axial compressor blade design. In ASME, editor, *ASME Turbo Expo 2006*, number GT2006-90420, Januar 2006.

[73] A. Schmitz. *Multifidelity-Optimierungsverfahren für Turbomaschinen.* PhD thesis, Ruhr-Universität Bochum. Noch nicht veröffentlicht.

[74] T. Lengyel-Kampmann. *Vergleichende aerodynamische Untersuchungen von gegenläufigen und konventionellen Fanstufen für Flugtriebwerke.* Phd thesis, Ruhr-Universität Bochum, August 2015.

[75] N. Bons, X. He, C. A. Mader, and J. Martins. Multimodality in aerodynamic wing design optimization. In *AIAA AVIATION Forum 2017*, pages –, June 2017.

[76] J. Nocedal and S. Wright. *Numerical optimization.* Springer Science & Business Media, 2006.

[77] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[78] L. M. Rios and N. V. Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, 2013.

[79] H.-P. P. Schwefel. *Evolution and optimum seeking: the sixth generation.* John Wiley & Sons, Inc., 1993.

[80] D. W. Zingg, M. Nemec, and T. H. Pulliam. A comparative evaluation of genetic and gradient-based algorithms applied to aerodynamic optimization. *European Journal of Computational Mechanics/Revue Européenne de Mécanique Numérique*, 17(1-2):103–126, 2008.

[81] F. Eulitz. *Numerische Simulation und Modellierung der instationären Strömung in Turbomaschinen.* PhD thesis, Ruhr-Universität Bochum, 2000.

[82] D. Nürnberger. *Implizite Zeitintegration für die Simulation von Turbomaschinenströmungen.* PhD thesis, Ruhr-Universität Bochum, Juni 2004.

[83] E. Kügeler. *Numerisches Verfahren zur genauen Analyse der Kühleffektivität filmgekühlter Turbinenschaufeln.* PhD thesis, Ruhr-Universität Bochum, 2005.

[84] A. P. Wheeler, R. D. Sandberg, N. D. Sandham, R. Pichler, V. Michelassi, and G. Laskowski. Direct numerical simulations of a high pressure turbine vane. In *ASME Turbo Expo 2015*, pages V02AT38A027–V02AT38A027. American Society of Mechanical Engineers, 2015.

[85] P. Tucker. Computation of unsteady turbomachinery flows: Part 1-progress and challenges. *Prog. Aerosp. Sci.*, 47(7):522 – 545, 2011.

[86] P. Tucker. Computation of unsteady turbomachinery flows: Part 2 -les and hybrids. *Prog. Aerosp. Sci.*, 47(7):546 – 569, 2011.

[87] J. Denton and U. Singh. Time marching methods for turbomachinery flow calculations. In *VKI Lecture Series 1979-7.* von Karman Institute., 1979.

[88] L. He. Fourier methods for turbomachinery applications. *Progress In Aerospace Sciences*, 46(8):329–341, November 2010.

[89] K. C. Hall, J. P. Thomas, and W. S. Clark. Computation of unsteady nonlinear flows in cascades using a harmonic balance technique. *AIAA Journal*, 40(5):879–886, May 2002.

[90] K. C. Hall, K. Ekici, J. P. Thomas, and E. H. Dowell. Harmonic balance methods applied to computational fluid dynamics problems. *International Journal of Computational Fluid Dynamics*, 27(2):52–67, 2013.

[91] C. Frey, G. Ashcroft, H.-P. Kersken, and C. Voigt. A harmonic balance technique for multistage turbomachinery applications. In *ASME Turbo Expo 2014*, 2014.

[92] G. Ashcroft, C. Frey, and H.-P. Kersken. On the development of a harmonic balance method for aeroelastic analysis. In *6th European Conference on Computational Fluid Dynamics (ECFD)*, 2014.

[93] A. Engels-Putzka and C. Frey. Adjoint harmonic balance method for forced response analysis in turbomachinery. In *Proceedings of the VI International Conference on Coupled Problems in Science and Engineering*, pages 645–656, 2015.

[94] A. Engels-Putzka and C. Frey. Sensitivity analysis for forced response in turbomachinery using an adjoint harmonic balance method. In *VII European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS)*, 2016.

[95] A. Engels-Putzka and C. Frey. Adjoint process chain for forced response analysis using a harmonic balance method. In *Proceedings of 12th European Conference on Turbomachinery Fluid dynamics & Thermodynamics*, 2017.

[96] C. Morsbach. *Reynolds stress modelling for turbomachinery flow applications*. PhD thesis, TU Darmstadt, 2016.

[97] D. C. Wilcox. Reassessment of the scale-determining equation for advanced turbulence models. *AIAA Journal*, 26(11):1299–1310, November 1988.

[98] M. Kato and B. E. Launder. The modeling of turbulent flow around stationary and vibrating square cylinders. In *9th Symposium on Turbulent Shear Flows*, pages 10.4.1–10.4.6, 1993.

[99] J. Bardina, J. H. Ferziger, and R. S. Rogallo. Effect of rotation on isotropic turbulence: computation and modelling. *J. Fluid Mech.*, 154:321–336, 1985.

[100] D. Kozulovic. Application of a multimode transition model to turbomachinery flows. In K. D. Papailiou, F. Martelli, and M. Manna, editors, *7th European Conference on Turbomachinery (ETC)*. National Technical University of Athens, March 2007.

[101] F. R. Menter, R. B. Langtry, S. R. Likki, Y. B. Suzen, P. G. Huang, and S. Volker. A correlation-based transition model using local variables—part I: Model formulation. *Journal of Turbomachinery*, 128(3):413–422, 2006.

[102] K. Becker, K. Heitkamp, and E. Kügeler. Recent progress in a hybrid-grid CFD solver for turbomachinery flows. In *V European Conference on Computational Fluid Dynamics (ECFD)*, 2010.

[103] P. L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357–372, 1981.

[104] B. van Leer. Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. *J. Comput. Phys.*, 32(1):101–136, 1979.

[105] A. Harten. High resolution schemes for hyperbolic conservation laws. *Journal of Computational Physics*, 49(2):357–393, 1983.

[106] J. D. Denton. The calculation of three-dimensional viscous flow through multistage turbomachines. *Journal of Turbomachinery*, 114(1):18–26, January 1992.

[107] M. B. Giles. Nonreflecting boundary conditions for Euler calculations. *AIAA Journal*, 28(12):2050–2058, 1990.

[108] C. Voigt, J. Wellner, C. Morsbach, and E. Kügeler. Conquer the terabyte-scale: Post-processing of high resolution unsteady CFD data for turbomachinery analysis. In *VI European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS)*, Vienna, Austria, September 2012.

[109] W. Yamazaki, M. P. Rumpfkeil, and D. J. Mavriplis. Design optimization utilizing gradient/Hessian enhanced surrogate model. *AIAA Paper*, 4363:2010, 2010.

[110] B. Christianson. Reverse accumulation and attractive fixed points. *Optimization Methods and Software*, 3(4):311–326, 1994.

[111] R. Giering and T. Kaminski. Recipes for adjoint code construction. *ACM Transactions on Mathematical Software*, 24(4):437–474, 1998.

[112] M. Giles. An extended collection of matrix derivative results for forward and reverse mode automatic differentiation. In *5th International Conference on Automatic Differentiation*. Unspecified, 2008.

[113] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys (CSUR)*, 23(1):5–48, 1991.

[114] J. R. Martins, P. Sturdza, and J. J. Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software (TOMS)*, 29(3):245–262, 2003.

[115] J. Martins, P. Sturdza, and J. J. Alonso. The connection between the complex-step derivative approximation and algorithmic differentiation. *AIAA paper*, 921:2001, 2001.

[116] L. Hascoët and V. Pascual. The Tapenade Automatic Differentiation tool: Principles, Model, and Specification. *ACM Transactions On Mathematical Software*, 39(3), 2013.

[117] D. Vandevoorde and N. M. Josuttis. *C++ Templates*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[118] T. Veldhuizen. Expression templates. *C++ Report*, 7(5):26–31, 1995.

[119] E. T. Phipps, R. A. Bartlett, D. M. Gay, and R. J. Hoekstra. Large-scale transient sensitivity analysis of a radiation-damaged bipolar junction transistor via automatic differentiation. *Advances in Automatic Differentiation, Springer*, pages 351–362, 2008.

[120] R. J. Hogan. Fast reverse-mode automatic differentiation using expression templates in C++. *ACM Trans. Math. Softw.*, 40:26:1–26:16, 2014.

[121] U. Naumann. Dco short introduction. Technical report, NAG, 2015.

[122] M. Sagebaum, T. Albring, and N. R. Gauger. High-performance derivative computations using codipack. *arXiv preprint arXiv:1709.07229*, 2017.

[123] A. Griewank. Some bounds on the complexity of gradients, jacobians, and hessians. *Complexity in Nonlinear Optimization*, pages 128–161, 1993.

[124] M. Sagebaum, E. Özkaya, N. R. Gauger, J. Backhaus, C. Frey, S. Mann, and M. Nagel. Efficient Algorithmic Differentiation Techniques for Turbo-machinery Design. In *AIAA-Paper 2017-3998*, 2017.

[125] F. Courty, A. Dervieux, B. Koobus, and L. Hascoet. Reverse automatic differentiation for optimum design: from adjoint state assembly to gradient computation. *Optimization Methods and Software*, 18(5):615–627, 2003.

[126] S. Nadarajah and A. Jameson. Studies of the continuous and discrete adjoint approaches to viscous automatic aerodynamic shape optimization. In *15th AIAA Computational Fluid Dynamics Conference*, page 2530, 2001.

[127] M. B. Giles. Discrete adjoint approximations with shocks. In *Hyperbolic problems: theory, numerics, applications*, pages 185–194. Springer, 2003.

[128] C. Frey, G. Ashcroft, J. Backhaus, E. Kügeler, and J. Wellner. Adjoint-based flow sensitivity analyis using arbitrary control surfaces. In *Proceedings of ASME-GT2011*, 2011.

[129] C. Frey, A. Engels-Putzka, and E. Kügeler. Adjoint boundary conditions for turbomachinery flows. In *European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS)*, 2012.

[130] Y. Saad. *Iterative methods for sparse linear systems. 2nd ed.* SIAM Society for Industrial and Applied Mathematics, Philadelphia., 2003.

[131] M. C. G. Hall and D. G. Cacuci. Physical interpretation of the adjoint functions for sensitivity analysis of atmospheric models. *Journal of the Atmospheric Sciences*, 40(10):2537–2546, 1983.

[132] S. Shankaran, A. Marta, P. Venugopal, B. Barr, and Q. Wang. Interpretation of adjoint solutions for turbomachinery flows. In *ASME Turbo Expo 2012*, number 44748, pages 2229–2242, 2012.

[133] A. Engels-Putzka and C. Frey. Adjoint mesh deformation and adjoint-based sensitvitities with respect to boundary values. In *VI European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS)*, pages 1789–1808, 2012.

[134] C. Frey, G. Ashcroft, H.-P. Kersken, and C. Weckmüller. Advanced numerical methods for the prediction of tonal noise in turbomachinery — Part II: Time-linearized methods. *Journal of Turbomachinery*, 136(2):021002, 2013.

[135] J. Backhaus, A. Schmitz, C. Frey, S. Mann, M. Nagel, M. Sagebaum, and N. R. Gauger. Application of an algorithmically differentiated turbomachinery flow solver to the optimization of a fan stage. In *AIAA-Paper 2017-3997*, 2017.

[136] B. Stroustrup. C and C++: Case studies in compatibility. *C/C++ Users Journal*, 20(9):22–31, 2002.

[137] A. Walther and A. Griewank. Getting started with adol-c. *Combinatorial Scientific Computing*, pages 181–202, 2012.

[138] P. M. Duvall, S. Matyas, and A. Glover. *Continuous integration: improving software quality and reducing risk.* Pearson Education, 2007.

[139] P. D. Hovland. *Automatic differentiation of parallel programs.* PhD thesis, University of Illinois at Urbana-Champaign, USA, 1997.

[140] M. Schanen, U. Naumann, L. Hascoët, and J. Utke. Interpretative adjoints for numerical simulation codes using MPI. In *Proceedings of ICCS2010*, volume 1, pages 1825–1833. Elsevier, 2010.

[141] J. Utke, L. Hascoët, P. Heimbach, C. Hill, P. Hovland, and U. Naumann. Towards adjoinable MPI. In *IEEE International Symposium on Parallel & Distributed Processing, 2009. IPDPS 2009*, pages 1–8. IEEE, 2009.

[142] W. J. Schroeder, B. Lorensen, and K. Martin. *The visualization toolkit: an object-oriented approach to 3D graphics*. Kitware, 2004.

[143] A. Griewank, C. Bischof, G. Corliss, A. Carle, and K. Williamson. Derivative convergence for iterative equation solvers. *Optimization methods and software*, 2(3-4):321–355, 1993.

[144] R. Thormann and M. Widhalm. Linear-frequency-domain predictions of dynamic-response data for viscous transonic flows. *AIAA journal*, 2013.

[145] Q. Rendu, M. Philit, S. Labit, J. Chassaing, Y. Rozenberg, S. Aubert, and P. Ferrand. Timelinearized and harmonic balance navier-stokes computations of a transonic flow over an oscillating bump. In *11th International Symposium on Unsteady Aerodynamics, Aeroacoustics & Aeroelasticity of Turbomachines*, 2015.

[146] M. S. Campobasso and M. B. Giles. Effects of flow instabilities on the linear analysis of turbomachinery aeroelasticity. *Journal of propulsion and power*, 19(2):250–259, 2003.

[147] M. S. Campobasso and M. B. Giles. Stabilization of linear flow solver for turbomachinery aeroelasticity using recursive projection method. *AIAA journal*, 42(9):1765–1774, 2004.

[148] R. P. Dwight and J. Brezillon. Efficient and robust algorithms for solution of the adjoint compressible navier–stokes equations with applications. *International journal for numerical methods in fluids*, 60(4):365–389, 2009.

[149] T. Albring, T. Dick, and N. R. Gauger. Assessment of the Recursive Projection Method for the Stabilization of Discrete Adjoint Solvers. In *AIAA-Paper 2017-3664*, 2017.

[150] S. Xu, D. Radford, M. Meyer, and J.-D. Müller. Stabilisation of discrete steady adjoint solvers. *Journal of Computational Physics*, 299:175 – 195, 2015.

[151] J. Krakos and D. Darmofal. Effect of small-scale output unsteadiness on adjoint-based sensitivity. *AIAA Journal*, 48(11):2611–2623, 2010.

[152] J. Krakos, Q. Wang, S. Hall, and D. Darmofal. Sensitivity analysis of limit cycle oscillations. *Journal of Computational Physics*, 231(8):3228–3245, 2012.

[153] Q. Wang, R. Hu, and P. Blonigan. Least squares shadowing sensitivity analysis of chaotic limit cycle oscillations. *Journal of Computational Physics*, 267:210–224, 2014.

[154] A. Engels-Putzka, J. Backhaus, and C. Frey. Forced response sensitivity analysis using an adjoint harmonic balance solver. In *ASME Turbo Expo 2018*, 2018.

[155] D. E. Knuth. Structured programming with go to statements. *Computing Surveys*, 6:261–301, 1974.

[156] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21(4):345–383, 2001.

[157] M. Hampe. Zufall. In P. Sarasin and M. Sommer, editors, *Evolution: Ein interdisziplinäres Handbuch*. Springer, 2010.

[158] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. The MIT Press, 2006.

[159] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Neural Networks, 1993., IEEE International Conference On*, pages 586–591. IEEE, 1993.

[160] C. R. Henderson. Best linear unbiased estimation and prediction under a selection model. *Biometrics*, pages 423–447, 1975.

[161] R. Zimmermann. Asymptotic behavior of the likelihood function of covariance matrices of spatial gaussian processes. *Journal of Applied Mathematics*, 2010, 2011.

[162] A. Tikhonov and V. Arsenin. Solutions of ill-posed problems,(translated from the russian). *Preface by translation editor Fritz John. Scripta Series in Mathematics (VH Winston & Sons/Wiley, Washington, DC/New York/Toronto/London, 1977)*, 1977.

[163] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[164] A. J. Keane. Statistical improvement criteria for use in multiobjective design optimization. *AIAA journal*, 44(4):879–891, 2006.

[165] T. Wagner, M. Emmerich, A. Deutz, and W. Ponweiser. On expected-improvement criteria for model-based multi-objective optimization. In *International Conference on Parallel Problem Solving from Nature*, pages 718–727. Springer, 2010.

[166] A. I. J. Forrester, A. J. Keane, and N. W. Bressloff. Design and analysis of "noisy"computer experiments. *AIAA journal*, 44(10):2331–2339, 2006.

[167] Flightpath 2050: Europe's vision for aviation. Europäische Kommission, 06 2011.

[168] T. C. Gmelin, G. Hüttig, and O. Lehmann. Zusammenfassende Darstellung der Effizienzpotenziale bei Flugzeugen unter besonderer Berücksichtigung der aktuellen Triebwerkstechnik sowie der absehbaren mittelfristigen Entwicklungen. *Bundesministerium für Umwelt, Naturschutz und Reaktorsicherheit. Berlin.*, 2008.

[169] H. Rick. *Gasturbinen und Flugantriebe. Grundlagen, Betriebsverhalten und Simulation*. Springer Berlin Heidelberg, 2013.

[170] MTU AeroEngines AG. Geschäftsbericht, 2007. Auch unter http://www.mtu.de/GB_2007_dt/10.0.html.

[171] P. Schimming, G. Schewe, S. Schmitt, R. Schnell, W. Neise, L. Enghardt, Y. Zhang, and L. Wallscheid. Experimental investigations on a CRISP-1m-model concerning its aeroacoustics, aeroelastics and aerodynamics. Final Project Report. BMBF LFT9601, German Aerospace Center (DLR), 2000.

[172] D. Goerke, A.-L. Le Denmat, T. Schmidt, F. Kocian, and E. Nicke. Aerodynamic and mechanical optimization of cf/peek blades of a counter rotating fan. In *ASME Turbo Expo 2012*, pages 21–33. American Society of Mechanical Engineers, 2012.

[173] J. A. Samareh. Survey of shape parameterization techniques for high-fidelity multidisciplinary shape optimization. *AIAA journal*, 39(5):877–884, 2001.

[174] J. Samareh. Aerodynamic shape optimization based on free-form deformation. In *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 4630, 2004.

[175] E. Arian and V. N. Vatsa. A preconditioning method for shape optimization governed by the Euler equations. *International Journal of Computational Fluid Dynamics*, 12(1):17–27, 1999.

[176] R. Jesudasan, X. Zhang, M. Gugala, and J.-D. Müller. CAD-free vs CAD-based parametrisation method in adjoint based aerodynamic shape optimization. In *VII European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS)*, 2016.

[177] C.-H. Wu. A general theory of three-dimensional flow in subsonic and supersonic turbomachines of axial-, radial, and mixed-flow types. Technical Report NACA-TN-2604, NASA Technical Report, 1952.

[178] L. Piegl and W. Tiller. *The NURBS book*. Springer Science & Business Media, 2012.

[179] A. Weber. 3D structured grids for multistage turbomachinery applications based on g3dmesh (1st. revision). Technical report, DLR, 2004.

[180] T. Lengyel-Kampmann, A. Bischoff, R. Meyer, and E. Nicke. Design of an economical counter rotating fan: comparison of the calculated and measured steady and unsteady results. In *ASME Turbo Expo 2012*, pages 323–336. American Society of Mechanical Engineers, 2012.

[181] S. Auriemma, M. Banovic, O. Mykhaskiv, H. Legrand, J.-D. Mueller, T. Verstraete, and A. Walther. Optimisation of a u-bend using a CAD-based adjoint method with differentiated CAD kernel. In *VII European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS)*, 2016.

[182] G. Dhondt. *The finite element method for three-dimensional thermomechanical applications*. John Wiley & Sons, 2004.

[183] D. Görke, A.-L. Le Denmat, T. Schmidt, F. Kocian, and E. Nicke. Aerodynamic and mechanical optimization of CF/PEEK blades of a counter rotating fan. In *ASME Turbo Expo 2012*, 2012.

Lebenslauf

Seit 11/2009	Wissenschaftlicher Mitarbeiter Deutsches Zentrum für Luft- und Raumfahrt Institut für Antriebstechnik, Köln Abteilung Numerische Methoden
10/2007 – 10/2009	Studentische Hilfskraft im Deutschen Zentrum für Luft- und Raumfahrt, Köln, Institut für Antriebstechnik Abteilung Fan- und Verdichter
10/2007 – 09/2009	Bergische Universität Wuppertal Masterstudiengang Informationstechnik Abschluss: Master of Science Absolventenpreis 2009 für ausgezeichnete Leistungen
09/2004 – 09/2007	Berufsakademie Mannheim Diplomstudiengang Informationstechnik Vertiefungsrichtung Ingenieurinformatik Abschluss: Diplom-Ingenieur (Berufsakademie)
09/2003 – 06/2004	Zivildienst
08/1994 – 07/2003	Allgemeine Hochschulreife Erich-Fried-Gesamtschule, Wuppertal